

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
28 March 2002 (28.03.2002)

PCT

(10) International Publication Number
WO 02/25570 A2

(51) International Patent Classification⁷: **G06F 19/00**

(21) International Application Number: **PCT/US01/42053**

(22) International Filing Date:
6 September 2001 (06.09.2001)

(25) Filing Language: **English**

(26) Publication Language: **English**

(30) Priority Data:
09/657,218 7 September 2000 (07.09.2000) **US**

(71) Applicant: **ARRAYEX, INC.** [US/US]; 2530 Meridian Parkway, Suite 3016, Durham, NC 27713 (US).

(72) Inventors: **WILBANKS, John, Thompson**; 8101 Reynard Road, Chapel Hill, NC 27516 (US). **SEGARAN, Suresh, Toby**; 700 Bolinwood, #16F, Chapel Hill, NC 27514 (US). **NABHAN, Antoun, Alexander**; 923 North Mangum Street, Durham, NC 27701 (US). **HESTER-BERG, Diane, Joyce**; 43 Prospect Street, Quincy, MA 02171 (US).

(74) Agent: **MYERS BIGEL SIBLEY & SAJOVEC**; P.O. Box 37428, Raleigh, NC 27627 (US).

(81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— without international search report and to be republished upon receipt of that report

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: **SYSTEMS, METHODS AND COMPUTER PROGRAM PRODUCTS FOR PROCESSING GENOMIC DATA IN AN OBJECT-ORIENTED ENVIRONMENT**

(57) Abstract: Systems, methods and computer program products process genomic data by providing a hierarchy of object superclasses and subclasses that represent genomic data, each object superclass/subclass including associated variables for the genomic data and methods that are performed on the genomic data, also referred to as an object library. Messages are sent between the hierarchy of superclasses and subclasses that represent genomic data, to thereby store, recall and analyze the genomic data. Messages may be sent between the hierarchy of object superclasses and subclasses that represent genomic data in response to a user request to store, recall or analyze the genomic data. These messages also may be sent in response to automated processes that run independent of user requests to store, recall or analyze the genomic data. The messages also may be sent in response to genomic data that is obtained by crawling a network such as the World Wide Web. Messages also may be sent between the hierarchy of object superclasses and subclasses that represent genomic data, to analyze the genomic data by finding patterns of gene expression levels among genes in experiments, and/or to analyze a gene expression signature for a pathology. An instance of at least one of the object superclasses and subclasses that represent genomic data also may be generated in response to receipt of raw genomic data. The raw genomic data may be received from a user, a process that is automatically run and/or a network crawler. Gene expression levels among genes in experiments that meet a predefined percentage change in expression level between the plurality of experiments may be displayed. The patterns of gene expression levels among the genes in the experiments also may be displayed as a function of gene families to which the plurality of genes in the plurality of experiments belong. The patterns of gene expression levels among the genes in the experiments also may be displayed as a scatter plot of gene expression levels among genes in two experiments, wherein at least some of the points in the scatter plot are surrounded by an error cloud that indicates an amount of error in the two experiments.

WO 02/25570 A2

**SYSTEMS, METHODS AND COMPUTER PROGRAM PRODUCTS FOR
PROCESSING GENOMIC DATA IN AN OBJECT-ORIENTED
ENVIRONMENT**

Field of the Invention

The present invention relates to bioinformatics, and more particularly to systems, methods and computer program products for processing genomic data.

5

Background of the Invention

The sequence of the human genome can provide a valuable medical resource. Unfortunately, in order to use this vast amount of sequence information to develop new medical applications, a more sophisticated understanding of gene function may be needed. In a sense, genome sequencing efforts are yielding a large quantity of
10 nouns, with the verbs and grammar yet to be fully discovered. Accordingly, much research effort has been focused on the interpretation of this vast amount of sequence information. This can result in a better understanding of the roles that genes and proteins play in biochemical pathways, and can thereby provide an understanding of
15 the mechanisms of disease.

These advances in genomics may also allow the drug discovery process to be transformed through rapid and efficient discovery of new drug targets in model organisms and human cells. In particular, drugs may target proteins or other compounds within each cell that are known to play a part in the metabolic pathway of
20 a disease. When these targets are identified, users may test many compounds against them. Based on the reaction of the target to the compound, a determination may be made as to whether a potential drug candidate is likely to be successful.

Thus, genomics has given rise to a variety of methodologies that are being used to discover new target molecules and therapeutic approaches. For example, the
25 discovery of new targets may be facilitated by comparing the DNA sequence of the potential target with that of known targets. If the DNA is similar, the proteins which

result may also be similar, suggesting that they will respond similarly to therapies. This approach also may be used to identify which molecular target in humans is likely to be analogous to a target previously identified in an animal model. Users also can identify targets by determining which genes are responsible for a given disease.

5 Genomics also can identify genetic variations which are a major component, either as a cause or as an effect, of diseases, such as cancer, diabetes and cardiovascular disease. Disease risks can be identified by monitoring variations in responsible genes. This may be done by analyzing mutations of a single nucleotide base, referred to as a Single Nucleotide Polymorphism (SNP). Unfortunately, 10 although SNPs may potentially indicate which drug will be best for a given individual, SNP analysis may need large-scale human studies to establish these useful associations. This may make SNP an expensive and difficult process, which also may be inaccurate, non-automated, inflexible and/or slow, depending on the implementation.

15 Genomics companies may focus on generating large amounts of DNA sequence data. Unfortunately, without knowledge of the gene's functions, the DNA sequence data for a gene may be insufficient to materially impact the drug development process. Moreover, associations between DNA sequence and detailed cellular function may be complex, and may be generally unknown. Accordingly, 20 detailed measurements of the actual biological functioning of the cell at a molecular level may be important to identify the best targets and illuminate mechanisms of disease.

Many approaches have been developed that can address these needs by monitoring changes in the levels of certain cellular components. One approach, 25 referred to as expression profiling, monitors the level of messenger RNA (mRNA) for each gene within a cell. Expression profiling technologies can monitor tens of thousands of genes. Monitoring of tens of thousands of genes may be performed by arranging shorter, single-stranded DNA pieces, called oligonucleotides, in a dense grid on a substrate, such as a glass surface. This grid is known as a microarray. An 30 oligonucleotide in a microarray may bind to the mRNA of a specific gene, to thereby provide an indication of that gene's expression level.

A second approach, referred to as "proteomics", monitors the level of protein expressed by each gene within a cell. Proteomics measurements may be obtained by separating a mix of proteins in a cell, by dragging the proteins through a resistive

substance, such as a gel, so that proteins of different sizes and properties end up in different spots on the gel. This microarray of spots then are analyzed, to thereby allow the monitoring of protein levels within the cell.

Unfortunately, expression profiling, proteomics and/or other genomic data
5 generation techniques may generate massive amounts of genomic data. For example, a microarray may yield a data file with several thousand lines, each of which represents a single spot on the array, and which may contain two numbers for the brightness of the scanner's reflection for that spot expressed using two wavelengths of light to correspond with two types of dye. In order to obtain useful information from
10 this massive amount of data, a scientist may engage in weeks or more of intensive analysis. Moreover, scientists may find that they cannot successfully analyze the data, so that microarray experiments may not yield useful results, because the useful results remain hidden in the massive amount of genomic data.

In an attempt to analyze these massive amounts of genomic data, the field of
15 bioinformatics has emerged. See, for example, U.S. Patent No. 5,876,933 to Perlin entitled *Method and System for Genotyping*; U.S. Patent No. 5,930,154 to Thalhammer-Reyero entitled *Computer-Based System and Methods for Information Storage, Modeling and Simulation of Complex Systems Organized in Discrete Compartments in Time and Space*; and U.S. Patent No. 5,966,711 to Adams entitled
20 *Autonomous Intelligent Agents for the Annotation of Genomic Databases*.

Unfortunately, notwithstanding these and other advances, it may be difficult for genomic data processing systems, methods and computer program products to cope with the massive amounts of data that are generated using microarrays. Databases loaded with the results of microarray experiments may quickly grow to sprawling
25 size. Moreover, the analysis tools may not scale well, so that a multivariable search may use excessive time and/or data processing resources to perform analysis.

Genomic data software analysis tools also are becoming available. See for example, *Software for MicroArray Gene Expression Analysis*,
http://www.edp.unil.ch/biocomputing/array/software/MicroArray_Software.html.
30 Two examples of software tools now will be described. These tools use image analysis and clustering/sorting.

Image analysis can provide a pictorial representation, on a computer screen, of the fluorescent images from single color or two color fluorescent hybridizations. To increase throughput, arrayers may use two printing tips. The image of a single tip,

called a sector, includes rectangularly- or hexagonally-arranged DNA elements, referred to as "spots". In order to extract data from the microarray images, it may be desirable to identify the location of each of the spots. Image analysis software can provide a graphical environment for performing this process, referred to as "gridding", and can simplify and semi-automate the process. The user may be presented with a grid of spots, also referred to as a "lite-briteTM" presentation, and can access each spot to reveal the gene accession number associated with each spot on the array. The intensity and hue of each spot relative to the other spots can tell the user which genes are more highly expressed relative to the others, and by how much. Unfortunately, it may be exceedingly difficult to detect patterns from the lite-brite image that confronts the user. Image analysis software includes the GenePix Pro 3.0 microarray analysis software, marketed by Axon Instruments, Inc., Foster City, California, www.axon.com.

Clustering/sorting software also may be used to assemble a set of items such as genes or arrays into a tree, where items are joined by very short branches if they are very similar to each other, and by very large branches as the similarity decreases. One form of clustering is the Self-Organizing Map (SOM). Self-organizing maps are described in a publication by Tamayo et al. entitled *Interpreting Patterns of Gene Expression With Self-Organizing Maps: Methods and Application to Hematopoietic Differentiation*, Proceedings of the National Academy of Science, USA, Vol. 96, March 1999, pp. 2907-2912; in a publication by Golub et al. entitled *Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring*, Science, Vol. 286, October 15, 1999, pp. 531-537; in a publication by Swenson et al. entitled *The Moulage System (Making the Right Impression)*, Cellworks Project, University of Washington, <http://cellworks.washington.edu/pub/cellworksAPI/index.html>; in a press release entitled "Self-Organizing Maps" Help Analyze Thousands of Genes, <http://www.eurekalert.org/releases/wimit-som031599.html>; and at the Stanford University site <http://rana.stanford.edu/software/>.

As described in the above-cited Tamayo et al. publication, self-organizing maps may be constructed by selecting a geometry of nodes N . The nodes are mapped into k -dimensional space, initially at random and then iteratively adjusted. Each iteration involves randomly selecting a data point P and moving the nodes in the direction of P . The closest node N_P is moved the most, whereas other nodes are

5 moved by smaller amounts depending on their distance from N_p in the initial geometry. In this fashion, neighboring points in the initial geometry tend to be mapped to nearby points in k-dimensional space. The process continues for 20,000-50,000 iterations. Unfortunately, by using between 20,000 and 50,000 iterations over all of the thousands of genes and multiple experiments, the generation of self-organizing maps may consume excessive time and/or data processing resources.

Summary of the Invention

10 Embodiments of the present invention provide systems, methods and computer program products that process genomic data by providing a hierarchy of object superclasses and subclasses that represent genomic data, each object superclass/subclass including associated variables for the genomic data and methods that are performed on the genomic data. Messages are sent between the hierarchy of superclasses and subclasses that represent genomic data, to thereby store, recall and
15 analyze the genomic data. The hierarchy of object superclasses and subclasses that represent genomic data also will be referred to herein as a "library". As is well known to those having skill in the art, the genomic data may be obtained through large sets of experiments in laboratories including microarray analysis ("DNA chips"), Polymerase Chain Reaction (PCR) a variety of blots and/or other techniques. As also is well
20 known to those having skill in the art, an object defines a data structure and a set of operations or functions that can access that data structure. The data structure may be represented as a frame that includes variables or attributes of the data in the frame. Each operation or function that can access the data structure is called a "method". Each defined object superclass and subclass may be manifested in a number of
25 "instances", with each instance containing the particular data structure for a particular example of the object.

Object-oriented programming systems, methods and computer program products can provide two characteristics which can allow flexible and scalable programs to be developed. These characteristics are referred to as "encapsulation" and "inheritance". In particular, the frame of variables is encapsulated by its methods
30 (functions), so that all access to the frame is handled by the surrounding methods. Data independence thereby may be provided because the object's data structure is accessed only by its methods. Data integrity thereby may be obtained. The "inheritance" property of object-oriented programming systems, methods and

computer program products can allow new objects to be described by how they differ from preexisting objects, so that entirely new programs need not be written to handle new types of data or functions. A subclass inherits the frame and methods of its superclass.

5 In an object-oriented system, method or computer program product, a high level routine can request an object to perform one of its methods by sending the object a "message" telling the object what to do. The receiving object responds to the message by choosing the method that implements the message name, executing this method, and then returning control to the high level routine, along with the results of
10 the method.

 Hierarchies of object superclasses and subclasses that represent genomic data, according to embodiments of the present invention, can provide a library of software code objects that are executable and can be used to construct algorithms and/or functional software codes. This contrasts sharply from a database that merely uses
15 object orientation to store data, such as the genome database which merely represents database rows in an object-oriented environment, but which may not be used as functional components of software.

 By providing a hierarchy of object superclasses and subclasses that represent genomic data, this hierarchy can be used to build algorithms, such as pattern finding
20 algorithms that described in detail below, and can also be used by a user to write software code that can mimic cellular process in its executable code. Moreover, by providing a hierarchy of object superclasses and subclasses that represent genomic data, the massive amounts of genomic data may be stored, recalled and analyzed efficiently. New data sets may be added by creating new instances, and new types of
25 data may be added by creating new object superclasses and/or subclasses in the hierarchy.

 In some embodiments, the hierarchy of object superclasses and subclasses that represent genomic data may include a genetic object superclass and at least one of an experiment subclass, a gene expression subclass, a pathology subclass, a mutation
30 subclass, a genomic DNA subclass, a structure subclass, a protein subclass and a pathway subclass of the genetic object superclass. In other embodiments, an experiment superclass may be provided that represents at least one microarray experiment. An experiment subclass of the experiment set superclass also may be provided that represents a single microarray experiment. A gene expression subclass

of the experiment subclass may be provided that represents a single point in a microarray experiment.

In yet other embodiments, a gene superclass may be provided that represents at least one gene. A first subclass of the gene superclass may be provided that
5 represents genes that express under predetermined conditions. A second subclass of the first subclass may be provided that represents genes that express under the predetermined conditions and use a predetermined protein to express. A third subclass of the second subclass may be provided that represents genes that express under the predetermined conditions, that use the predetermined protein to express and
10 that use a predetermined sequence.

In still other embodiments, a genetic object superclass may be provided, along with a pathology subclass of the genetic object superclass and a cancer subclass of the pathology subclass. Other disease subclasses of the pathology subclass may be provided. In yet other embodiments, a genetic object superclass, a genomic DNA
15 subclass of the genetic object superclass, a gene subclass of the genomic DNA subclass and an expressed sequence tag subclass of the genomic DNA subclass may be provided. In still other embodiments, a genetic object superclass, a protein subclass of the genetic object superclass, a receptor subclass of the protein subclass and an ion-gated receptor subclass of the receptor subclass may be provided.

20 According to other embodiments of the invention, messages may be sent between the hierarchy of object superclasses and subclasses that represent genomic data in response to a user request to store, recall or analyze the genomic data. In yet other embodiments, these messages may be sent in response to automated processes that run independent of user requests to store, recall or analyze the genomic data. In
25 other embodiments, the messages may be sent in response to genomic data that is obtained by crawling a network such as the World Wide Web.

In other embodiments, messages are sent between the hierarchy of object superclasses and subclasses that represent genomic data, to analyze the genomic data by finding patterns of gene expression levels among at least one gene in at least one
30 experiment. In other embodiments, messages are sent to analyze a gene expression signature for a pathology.

In still other embodiments, an instance of at least one of the object superclasses and subclasses that represent genomic data is generated in response to receipt of raw genomic data. The raw genomic data may be received from a user,

from a process that is automatically run and/or from a network crawler. At least one filter may be provided, each of which converts the raw genomic data from a predetermined source into an instance of at least one of the object superclasses and subclasses that represent genomic data. A selected one of the filters may be applied to the raw genomic data based upon the source of the raw genomic data.

Other embodiments of the invention can display patterns of gene expression levels among at least one gene in at least one experiment that meets a predefined percentage change in expression level between the plurality of experiments. Other embodiments can display the patterns of gene expression levels among the plurality of genes in the plurality of experiments as a function of gene families to which the plurality of genes in the plurality of experiments belong. Yet other embodiments display the patterns of gene expression levels among the at least one gene in the at least one experiment as a scatter plot of gene expression levels among at least one gene in two experiments, wherein at least some of the points in the scatter plot are surrounded by an error cloud that indicates an amount of error in the two experiments.

Embodiments of the invention that analyze gene expression data for at least one gene and for at least one experiment can slice the gene expression data for the at least one gene and for the at least one experiment into a plurality of slices that define ranges of gene expression changes for the plurality of genes among the plurality of experiments. For each gene, a determination is made as to which of the slices each of the plurality of experiments belongs. The slicing may be performed by obtaining a percentage tolerance that defines the ranges of gene expression changes for the plurality of genes among the plurality of experiments. In embodiments of the invention, in order to determine to which of the slices each of the plurality of experiments belongs, a difference of a gene expression level is determined between a present experiment and a previous experiment, the difference of gene expression level is divided by the percentage tolerance and a value is obtained therefrom. The value is appended to the gene profile of the gene. The determining, dividing and appending is performed for each of the plurality of experiments.

In other embodiments of the invention, a pathology is diagnosed from gene expression test results for a test group having the pathology and from gene expression test results for a control group that is free of the pathology. The gene expression test results for a test group having the pathology and the gene expression test results for a control group that is free of the pathology are comparatively analyzed, to obtain a

listing of likely candidate genes and associated confidence levels. The listing of likely candidate genes is stored in an object-oriented library having a pathology object class, as an instance of a subclass of the pathology object class for the pathology. The analysis may be performed by calculating a mean and a standard deviation of the gene expression test results for a first gene for the first test group and for the control group, calculating a t-value for a hypothesis that the means for the first group and for the control group are equal, and determining a confidence value for the t-value that is calculated. If the hypothesis that the means for the first group and for the control group are equal is rejected, the first gene is added as a likely candidate for the pathology. The calculating of a mean and standard deviation, calculating a t-value and determining a confidence value are performed for remaining ones of the genes in the test group and in the control group, to thereby obtain a listing of likely candidate genes and associated confidence levels.

According to yet other embodiments of the present invention, new genomic data objects may be created for a genomic object library that contains a hierarchy of object superclasses and subclasses that represent genomic data, each including associated variables for the genomic data and methods that are performed on the genomic data. A list of genomic databases that are external to the genomic object library is provided. A first database in the list of genomic databases is reviewed to determine if new information has been added thereto. The new information is obtained if the new information has been added to the first database. The new information is converted into an instance of one of the object superclasses or subclasses and is stored in the genomic object library. The remaining databases in the list of genomic databases that are external of the genomic object library also are reviewed periodically, and new information is obtained, converted into an instance and stored as was described above. In some embodiments of the invention, the review may be performed in response to a user query of the genomic object library for a genomic data object that is not in the genomic object library. In other embodiments, the reviewing may be performed periodically, independent of user queries of the genomic object library. In yet other embodiments, new genomic databases may be added to the list of genomic databases that are external of the genomic object library. The new genomic database may be obtained and converted into a second instance of one of the object superclasses and subclasses and stored in the genomic object library. In some embodiments, adding a new genomic database may be performed in response

to a user query of the genomic object library for a genomic data object that is not in a genomic object library. In other embodiments, adding may be performed periodically, independent of user queries of the genomic object library.

According to yet other embodiments of the invention, gene expression data is displayed by sorting sets of gene expression data for at least one experiment based on a gene family to which each gene belongs. The gene expression data for the plurality of experiments is displayed, grouped by the gene family. In some embodiments, the gene expression data may be sorted by obtaining a set of gene expression experimental results and a minimum change between gene expressions to be displayed, creating a table or other data structure that is indexed by gene family, obtaining a family name for a first gene that changes more than the minimum change across the set of gene expression results and storing, under the index of the family name that is obtained, an identification of the first gene and the gene expression level for the first gene that changes more than the minimum change across the set of gene expression results. Family names may be obtained and the identification may be stored under the index of the family name for remaining genes in the set of gene expression results that change by more than the minimum change, to thereby store in the table the gene expression results grouped by family name. In other embodiments, the gene expression data may be displayed by displaying an array of boxes, each of which includes a gene family name associated therewith, displaying an array of subboxes in each box, each subbox corresponding to a gene in the associated gene family, and displaying in each subbox an indicium of the associated expression level for the associated gene. The indicium may be a brightness that is in direct proportion to the associated expression level for the associated gene, a color, a design, a saturation level and/or other indicium.

In other embodiments of the invention, gene expression data for first and second experiments is displayed by displaying at least one point in a two-dimensional coordinate system, each point being located at coordinates that correspond to the gene expression data for a gene in the first and second experiments. An error cloud is displayed around at least some of the points that indicates an error in the gene expression data for the associated gene, in the first and second experiments. The error cloud may be generated by identifying a gene's error in the first experiment and the gene's error in the second experiment, and displaying the cloud centered about the point that is determined by the gene's expression in the first experiment and the gene's

expression in the second experiment, wherein the cloud has a height that is determined by the amount of the gene's error in the first experiment and a width that is determined by the amount of the gene's error in the second experiment. A three-dimensional error cloud, corresponding to three experiments, also may be generated and displayed.

Other embodiments of the invention can store genomic data in a genomic object library by obtaining genomic data from a source that is external to the library, the genomic data including an identification of the source. One of a plurality of object-generating modules is applied to the genomic data. The object-generating module is selected based on the identification of the source, to generate at least one object from genomic data, the object including associated variables for the genomic data and methods that are performed on the genomic data. The at least one object then is stored in an object-oriented library that includes a hierarchy of object superclasses and subclasses that represent genomic data. For example, the genomic data may be obtained from a public database, such as the National Institute of Health (NIH) database, that is external to the object-oriented database. The object-generating module may be instantiated based on the identification of the source, and the selected object-generating module may instantiate at least one object. The genomic data then is parsed to determine the associated variables for the genomic data and the methods that are performed on the genomic data.

Other embodiments of the present invention store genomic data in an object-oriented library that includes a hierarchy of object superclasses and subclasses that represent genomic data. A new genomic data set is obtained including at least one record, from a source that is external to the object-oriented library. A new software object is created for each record in the new genomic data set that has no counterpart software object in an earlier version of the genomic data set that is stored in the object-oriented library. The counterpart software object in the earlier version of the genomic set that is stored in the library is modified if the counterpart object exists in the earlier version of the genomic data set that is stored in the library. The new software object or the counterpart software object that is modified then is stored in the object-oriented library. In some embodiments, the new genomic data set is obtained in response to user input. In other embodiments, the new genomic data set is obtained in response to a database crawler that crawls a computer network such as the World Wide Web.

Yet other embodiments of the invention process genomic data by providing an object library that contains a hierarchy of object superclass instances and object subclass instances that represent genomic data. Each superclass instance and subclass instance includes associated variables for the genomic data and methods that are performed on the genomic data. An object generator converts raw genomic data into at least one of the object superclass instances or object subclass instances that represent the raw genomic data. At least one genomic analysis tool also is provided that is responsive to the object library to analyze the raw genomic data in the hierarchy of object superclass instances and object subclass instances using the methods. A user interface is responsive to user input to send the raw genomic data to the object generator and to initiate the plurality of genomic data analysis tools to analyze the genomic data in the hierarchy of object superclass instances and object subclass instances using the methods.

In some embodiments, the raw genomic data may be provided by a user. In other embodiments, a crawler searches a network such as the World Wide Web for raw genomic data and provides the raw genomic data that is obtained to the object generator. In yet other embodiments, a thread activates at least one of the genomic data analysis tools independent of user activation via a user interface. The user interface may include a client system that is responsive to user input to generate user requests and raw data and a server system that is responsive to the client system, the object generator and the plurality of genomic data analysis tools.

It will be understood that embodiments of the invention may be provided as systems, methods and/or computer program products.

Brief Description of the Drawings

Figure 1 is a block diagram of genomic data processing systems, methods and computer program products according to embodiments of invention.

Figures 2A and 2B, which when placed together as shown form Figure 2, illustrate hierarchies of object superclasses and subclasses that represent genomic data according to embodiments of the invention.

Figure 3 illustrates other hierarchies of object superclasses and subclasses that represent genomic data according to embodiments of the invention.

Figure 4 graphically illustrates slicing of gene expression data according to embodiments of the present invention.

Figure 5 is a flowchart of pattern finding systems, methods and computer program products according to embodiments of the invention.

Figure 6 is a user interface for pattern finding systems, methods and computer program products according to embodiments of the invention.

5 Figures 7-9 are data structures that may be used with pattern finding systems, methods and computer program products according to embodiments of the present invention.

Figure 10 is a display that may be produced by pattern finding systems, methods and computer program products according to embodiments of the invention.

10 Figure 11 is a flowchart of diagnosis systems, methods and computer program products according to embodiments of the present invention.

Figure 12 illustrates a user interface for diagnosis systems, methods and computer program products according to embodiments of the present invention.

15 Figure 13 is a flowchart of object generator systems, methods and computer program products according to embodiments of the present invention.

Figure 14 is a flowchart of automated systems, methods and computer program products according to embodiments of the invention.

Figure 15 is a flowchart of database crawler systems, methods and computer program products according to embodiments of the invention.

20 Figure 16 is a flowchart of experiment set producing systems, methods and computer program products according to embodiments of the present invention.

Figure 17 is a flowchart of gene family generating systems, methods and computer program products according to embodiments of the invention.

25 Figure 18 is a user interface for gene family generating systems, methods and computer program products according to embodiments of the invention.

Figure 19 is a display that may be produced by gene family generating systems, methods and computer program products according to embodiments of the present invention.

30 Figure 20 is a flowchart of operations that may be performed by systems, methods and computer program products that produce a scatter plot with an error cloud according to embodiments of the present invention.

Figure 21 illustrates a scatter plot with an error cloud that may be produced by systems, methods and computer program products according to embodiments of the invention.

Detailed Description of Preferred Embodiments

The present invention now will be described more fully hereinafter with reference to the accompanying drawings, in which embodiments of the invention are shown. This invention may, however, be embodied in many different forms and should not be construed as limited to the embodiments set forth herein. Rather, these
5 embodiments are provided so that this disclosure will be thorough and complete, and will fully convey the scope of the invention to those skilled in the art. Like numbers refer to like elements throughout.

10 The present invention is described below with reference to block diagrams and/or flowchart illustrations of methods, apparatus (systems) and/or computer program products according to embodiments of the invention. It is understood that each block of the block diagrams and/or flowchart illustrations, and combinations of blocks in the block diagrams and/or flowchart illustrations, can be implemented by
15 computer program instructions. These computer program instructions may be provided to a processor of a general purpose computer, special purpose computer, and/or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer and/or other programmable data processing apparatus, create means for implementing the
20 functions specified in the block diagrams and/or flowchart block or blocks.

These computer program instructions may also be stored in a computer-readable memory that can direct a computer or other programmable data processing apparatus to function in a particular manner, such that the instructions stored in the computer-readable memory produce an article of manufacture including instructions
25 which implement the function specified in the block diagrams and/or flowchart block or blocks.

The computer program instructions may also be loaded onto a computer or other programmable data processing apparatus to cause a series of operational steps to be performed on the computer or other programmable apparatus to produce a
30 computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide steps for implementing the functions specified in the block diagrams and/or flowchart block or blocks.

It should also be noted that in some alternative implementations, the functions noted in the blocks may occur out of the order noted in the flowcharts. For example,

two blocks shown in succession may in fact be executed substantially concurrently or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved.

5 In order to provide a complete description of embodiments of the invention, the present Detailed Description will begin with definitions of terms, continue with an overview of genomic data processing systems, methods and computer program products according to embodiments of the invention, and will provide a general description of the components thereof, as well as some general operational examples. Various embodiments of hierarchies of object superclasses and subclasses that
10 represent genomic data will then be described. Finally, embodiments of components of genomic data processing systems, methods and/or computer program products according to embodiments of the invention will be described in detail.

Definitions

15 As used herein, the following terms have the following meanings:

Genome – The total gene complement of a set of chromosomes found in higher life forms; or, the functionally similar, but simpler, linear arrangements found in bacteria and viruses. A genome may include, or be represented as, genomic DNA or cDNA and also may include mitochondrial DNA.

20 Gene – the functional unit of heredity. Each gene occupies a specific place (or locus) on a chromosome, is capable of reproducing itself exactly at each cell division, and is capable of directing the formation of a protein. The gene as a functional unit may consist of a discrete segment of a DNA molecule containing the proper number of purine (adenine and guanine) and pyrimidine (cytosine and thymine) bases in the
25 correct sequence to code the sequence of amino acids needed to form a specific peptide.

Gene Profile – a general term that typically describes the summary of the genes being expressed at one time in one cell.

Genomic Data - information on some or all of a genome.

30 Cancer – A general term used to indicate any of various types of malignant tumors, most of which invade surrounding tissues, may metastatize to several sites, and are likely to reoccur after attempted removal and to cause death unless adequately treated.

Receptor – a signaling molecule, such as Dopamine receptor, Glutamine receptor, Ethylene receptor, insulin receptor, photoreceptors, GABA(A) receptor, thrombin receptors, opioid receptor.

5 Pathology – The interpretation of diseases in terms of cellular operations; i.e. the way in which cells become unstable.

Mutation – a change in the character of a gene that is perpetrated in subsequent divisions of the cell in which it occurs, or, a change in the sequence of base pairs in the chromosomal molecule.

10 Protein – macromolecule consisting of long sequences of alpha-amino acids in peptide linkage involved in structures, hormones, enzymes, and essential life functions.

Structure – A tissue or formation made up of different or related parts; or, the specific connections of the atoms in a given molecule. Examples include muscle, nerve, skin, lung, liver, leaf, root, flower, stem and other tissues.

15 Genomic DNA – The DNA which makes up the entire chromosomal DNA of a life form.

Mitochondrial DNA – the DNA which rests in the mitochondria.

20 Gene Expression – the representation of genes that are being transcribed from the DNA and translated into proteins, but also in reference to a particular tissue, stage of development or combinations thereof.

Gene Expression Signature – summary of gene expression at one time in one profile – usually used in reference to pathology, but also in reference to the developmental stage of the organism, a response to stimuli such as drugs or environmental factors, tissue specificity, age, and disease progression.

25 Pathway – Any sequence of chemical reactions leading from one compound to another

Other terms that are used herein are well known to those having skill in the art and need not be described in detail herein.

30 Overview of Genomic Data Processing Systems, Methods and Computer Program Products

Figure 1 is a block diagram of genomic data processing systems, methods and computer program products according to embodiments of the invention. In the embodiments shown in Figure 1, the systems, methods and computer program

products may be divided into client systems, methods and computer program products ("client") 110, server systems, methods and computer program products 120 ("server") and back-end systems, methods and computer program products 130 ("back-end"). However, it will be understood by those having skill in the art that any or all of these systems, methods or computer program products may be integrated or altered in boundary. Thus, the client 110, server 120 and back-end 130 each may be contained in one or more enterprise, personal and/or pervasive computing devices, or may be combined on one or more enterprise, personal and/or pervasive computing devices. The client 110 may communicate with the server 120 over a network 118, which may be a wired and/or wireless, public and/or private, local and/or wide area network such as the World Wide Web and/or a sneaker network using portable media. Moreover, when the client 110 and server 120 are integrated, communication may take place via an Application Programming Interface (API) in one or more data processing systems. Similarly, communications between the server 120 and the back-end 130 may take place over a network 128 which may be a wired and/or wireless, private and/or public, local and/or wide area network such as the World Wide Web and/or a sneaker network using portable media. Moreover, when the server 120 and back-end 130 are integrated in a single data processing system, these communications may take place over an API.

A general description of the client 110, server 120 and back-end 130 now will be provided. It will be understood that in Figure 1, the connections between the components are illustrated as unidirectional, to follow the path of a request. It will be understood however, that these connections generally are bidirectional, so that a response may follow the opposite path from that shown.

Still referring to Figure 1, embodiments of the client 110 include a plurality of Web pages 112, and/or a plurality of Java applets 116, that execute on a client computer 114. The Web pages 112 and Java applets 116 may originate at the server 120, but may be downloaded and executed on the client computer 118. The Web pages 112 may be used to send user input requests to the server 120 via the communications connection 118. Specific examples of Web pages will be described in detail below. The Java applets 116 may be used to present results to a user of the client 110. Specific presentation of results according to embodiments of the invention, such as a gene family view and a scatter plot with error clouds, will be described in detail below.

A Web site is conventionally a related collection of Web files that includes a beginning file called a "home" page. From the home page, a visitor can access other files and applications at the Web site. A large Web site may utilize a number of servers, which may or may not be different and may or may not be geographically-dispersed. For example, the Web site of the International Business Machines Corporation (www.ibm.com) includes thousands of Web pages and files spread out over multiple Web servers in locations world-wide.

A Web server (also referred to as a Hyper-Text Transfer Protocol (HTTP) server) is a computer program that utilizes HTTP to serve files that form Web pages to Web clients. Exemplary Web servers are International Business Machines Corporation's family of Lotus Domino[®] servers, the Apache server (available from www.apache.org), and Microsoft's Internet Information Server (IIS), available from Microsoft Corporation, Redmond, Washington.

A Web client is a requesting program that also utilizes HTTP. A browser is an exemplary Web client for use in requesting Web pages and files from Web servers. Since browsers are exemplary Web clients, the term "browser" will be used herein to indicate any Web client. A Web server waits for a Web client to open a connection and to request a specific Web page or application. The Web server then sends a copy of the requested item to the Web client, closes the connection with the Web client, and waits for the next connection.

HTTP allows a browser to request a specific item, which a Web server then returns and the browser renders. To ensure that browsers and Web servers can interoperate unambiguously, HTTP defines the exact format of requests (HTTP requests) sent from a browser to a Web server as well as the format of responses (HTTP responses) that the Web server returns to the browser. Exemplary browsers include Netscape Navigator[®] (America Online, Inc., Dulles, VA) and Internet Explorer[®] (Microsoft Corporation, Redmond, WA). Browsers typically provide a graphical user interface for retrieving and viewing Web pages, applications, and other resources served by Web servers.

As is known to those skilled in this art, a Web page is conventionally formatted via a standard page description language such as HyperText Markup Language (HTML) and/or extensible markup language (XML), which typically contains text and can reference graphics, sound, animation, and/or video data. HTML provides for basic document formatting and allows a Web content provider to specify

anchors or hypertext links (typically manifested as highlighted text) to other servers. When a user selects (*i.e.*, activates) a particular hypertext link, a browser running on the user's client device reads and interprets an address, called a Uniform Resource Locator (URL) associated with the hypertext link, connects the browser with a Web
5 server at that address, and makes a request (*e.g.*, an HTTP request) for the file identified in the hypertext link. The Web server then sends the requested file to the client device which the browser interprets and renders within a display screen.

Still referring to Figure 1, the server 120 includes a server data processing system 122 that is linked to the client data processing system 114 via the
10 communications connection 118. The server computer 122, which may be a Web application server, includes Servlets 124 and Java Server Pages (JSP) pages 126. As is well known to those having skill in the art, JSPTM pages are dynamic Web pages that can be both platform-independent and server-independent. Servlets are pieces of JavaTM source code that can add functionality to a Web server in a manner similar to
15 the way applets can add functionality to a browser.

The Servlets 124 and JSP pages 126 may be invoked by the server computer 122 when requested by the client 110. They take a client request and pass it to the back-end 130 via communications connections 128. Although two communications
20 connections 128 are shown for functional purposes, a single communications connection may be provided. When the back-end 130 provides the results of the client request, the Servlets 124 and JSP pages 126 can respond to the client 110, either as a new Web page 112 or as data sent directly to a Java applet 116. Thus, the servlets 124 and JSP pages 126 act as middlemen between the client 110 and the back-end 130.

25 According to embodiments of the invention, the back-end 130 is configured as object-oriented programming systems, methods and/or computer program products including object-oriented database management systems, methods and computer program products. As is well known to those having skill in the art, in object-oriented programming, the primary focus is on the combination of data with functions. In
30 contrast, in non-object-oriented programming languages such as C, data and functions generally are represented as separate units. Object-oriented programming systems, methods and computer program products are composed of a large number of "objects". An object is a data structure and a set of operations or functions that can access that data structure. The data structure may be represented as a "frame". The

frame has many "slots", each of which contains an "attribute" of the data in the slot. The attribute may be a primitive (i.e. an integer or string) or an object reference which is a pointer to another object's instance or instances. Each operation (function) that can access the data structure is called a "method". Each defined object will usually be
5 manifested in a number of "instances" Each instance contains the particular data structure of a particular example of the object.

Object-oriented programming can provide two primary characteristics which allow flexible and reusable programs to be developed. These characteristics are referred to as "encapsulation" and "inheritance". In particular, the frame is
10 encapsulated by its methods (functions). A wall of code thus has been placed around each piece of data. Access to the frame is handled by the surrounding methods. Data independence is thereby provided because an object's data structure is accessed only by its methods. Only the associated methods know the internal data structure. This can ensure data integrity.

15 The "inheritance" property of object-oriented programming allows previously written programs to be broadened by creating new superclasses and subclasses of objects. New objects are described by how they differ from preexisting objects so that entirely new programs need not be written to handle new types of data or functions. Each subclass "inherits" the frame and methods of its superclass.

20 In an object-oriented system, a high level routine (messenger) requests an object to perform one of its methods by sending the object a "message" telling the object what to do. The receiving object responds to the message by choosing the method that implements the message name, executing this method and then returning control to the calling high level routine, along with the results of the method.

25 Object-oriented programming systems, methods and computer program products may be employed as database management systems, methods and computer program products which are capable of operating upon a large database, and which are expandable and adaptable. Hierarchies of object superclasses and subclasses that represent genomic data, according to embodiments of the present invention, can
30 provide a library of software code objects that are executable and can be used to construct algorithms and/or functional software code. Object libraries are described, for example at <http://www.opencascade.com/open/tov.html>. This contrasts sharply from a database that merely uses object orientation to store data, such as the genome database (<http://gdbwww.gdb.org/>) which merely represents database rows in an

object-oriented environment, but which may not be used as functional components of software. The database manager also may be organized as a set of objects with database management operations being performed by sending messages from one object to another. The target object performs the requested action on its attributes
5 using its methods.

Object-oriented processing systems, methods and computer program products are well known to those having skill in the art and are described, for example, in U.S. Patent 5,313,629 to Abraham et al., and in an article entitled *What is Object-Oriented Software? An Introduction*, <http://207.240.40.16/softinfo/objects.html>. Accordingly, a
10 detailed description of object-oriented systems, methods and computer program products and object-oriented database systems, methods and computer program products need not be provided herein.

Embodiments of the back-end 130 include an object library 136 that contains a hierarchy of object superclasses and subclasses that represent genomic data, each
15 including associated variables for the genomic data and methods that are performed on the genomic data. As such, the object library 136 is illustrated in Figure 1 as containing a hierarchy of objects, with each object being represented in a standard representation as a frame that is encapsulated within its methods. An object messenger 162 also is included that sends messages between the hierarchy of object
20 superclasses and subclasses that represent genomic data, to thereby store, recall and analyze the genomic data. Since the object messenger 162 may be connected to each of the blocks of the back-end 130, these connections are not illustrated for the sake of clarity.

Other embodiments of the back-end 130 may include an object library 136 that
25 contains a hierarchy of object superclass instances and object subclass instances that represent genomic data, each superclass instance and subclass instance including associated variables for the genomic data and methods that are performed on the genomic data. An object generator 138 converts raw genomic data into at least one of the object superclass instances or object subclasses instances that represent the raw
30 genomic data. A plurality of genomic data analysis tools 152 are responsive to the object library 136, to analyze the genomic data in the hierarchy of object superclass instances and object subclass instances. The server 120 and client 110 can provide a user interface that is responsive to user input, to send the raw genomic data to the object generator 138 and to initiate the plurality of genomic data analysis tools 152 to

analyze the genomic data in the hierarchy of object superclass instances and object subclass instances using the methods.

Other embodiments of the back-end 130 may add one or more of the other components that are illustrated in Figure 1, including an object provider 134, a database crawler 144, automated processes 146, a request broker 132, a tool provider 148 and filters 142. Each of the components of the back-end 130 of Figure 1 now will be described generally. Embodiments of these components will be described in greater detail below.

In particular, the request broker 132 can manage all requests to the back-end 130. When it receives a request for an analysis, it can obtain the necessary objects from the object provider 134, pass these objects to the tool provider 148, and obtain a result set which is passed back to the requester. The request broker 132 also may receive data which is to be saved in the object library 136, or may need to be saved as part of an analysis, in which case the request broker 132 saves the data in the object library 136 using the object provider 134.

The object provider 134 accesses the object library 136, retrieves object data, such as Binary Large Object (BLOB) data, deserializes the data and then returns the object. The object provider 134 also may provide caching of commonly used objects and logging of requested objects.

The object generator 138 obtains raw data along with a tag that identifies the source of the data. The raw data may be obtained from the request broker 132 or the database crawler 144. The object generator 138 uses the tag to look up, in a table or other indexed data structure, an appropriate filter 142 to use. For example, as shown in Figure 1, a generic array filter 142a and an NIH gene filter 142b may be provided. Other filters also may be provided.

After the filter 142 has been determined, the object generator 138 queries the filter for the type of objects it will return. The object generator 138 then calls the filter, passing it the raw data, and retrieves a group of objects from the filter 142. For each of these objects, the object generator checks if the object already is in the object library 136. If so, the newer object is used. The objects are then serialized and added to the library.

The tool provider 148 takes a group of objects and an analysis request that is obtained from the request broker 132, decides which tool 152 is appropriate for the requested analysis, and calls the tool 152 with the objects to retrieve a result. As

shown in Figure 1, a pattern finder tool 152a, a diagnosis engine tool 152b, and experiment set factory tool 152c are provided. Other tools also may be provided.

Automated processes 146 comprise a group of separate systems, methods and/or computer program products that can run continuously. They may contain a list
5 of many (for example, thousands) of analyses to try, and/or can be set to try random tests and look for patterns of interest. They can make requests for analysis to the request broker 132 and also can save interesting results to the library 136. The database crawler 144 also is a separate process or thread that can continuously run. It can scour databases and other sources of new information, for example using a
10 network such as the World Wide Web, and send this new information to the object generator 138.

Three general examples now will be provided to illustrate overall operation of embodiments of systems, methods and computer program products according to Figure 1. In the first example, a user attempts to find gene expression patterns. In the
15 second example, new genes are imported. In the third example, an automated treatment hypothesis is investigated.

More specifically, in the first example, the user desires to find patterns in gene expression data. The user selects the desired experiments and parameters and selects a "find patterns" selection at a user interface on the client computer 114. This causes
20 a Java applet 116 to send this information to the servlet 124. The servlet 124 constructs a request containing the experiment names, parameters and the request type, and sends it to the request broker 132. The request broker passes the experiment names to the object provider 134, and retrieves the relevant experiment objects from the library 136. The request broker then passes the objects to the tool provider 148,
25 along with the request type. The tool provider 148 determines that the request should be handled by the pattern finder 152a and passes along the objects. The pattern finder 152a returns a result set containing the resulting patterns. The tool provider 148 returns the result set to the request broker 132 and the request broker returns the result set to the servlet 124. The servlet 124 interprets the results and generates a new
30 HTML page with an applet 116 showing the results at the client data processing system 114.

In the second example, genes are imported into the system. For example, each month the NIH issues a batch update in a text-based format that contains information about newly discovered genes. The database crawler 144, preferably running

independently, connects to the NIH website and downloads the update. The database crawler 144 then passes the data to the object generator 138, along with a tag specifying "NIH NCBI Gene Database". The object generator 138 looks up this tag and determines that the NIH gene filter 142b is the appropriate filter 142 to use. The
5 Object Generator 138 queries the NIH gene filter 142b and is informed that it will return a group of objects of class "gene". The object generator 138 passes the raw data to the filter 142b and receives in response a group of gene class objects. The object generator 138 then queries the object library 136, to determine if any of these genes are already present. If so, it determines which record is newer and discards any
10 outdated objects. The gene class objects are then serialized and stored in the library 136.

In the third example, an automated process is started and asked to investigate a specific disease. A request is made to the request broker 132 by the client 110 for all the names of all expression experiments on patients in the library 136 who are
15 believed to have the disease, along with those known not to have the disease. The request broker 132 can retrieve this information from the object provider 134. All combinations of control and test group patients can be processed and the request broker can run the diagnosis engine 152b via the tool provider 148, to discover the differences. After performing this analysis, a list of likely genes involved with a
20 disease may be produced. The request broker can store this as a disease object in the library 136. The process then retrieves a list of experiments involving tissue being treated with a compound. It then makes a request to use the pattern finder 152a and uses the results to decide if this compound has the desired effect on the disease-related genes. For each compound that does have a possibly positive effect, the process can
25 update the disease object with this compound as a new possible treatment. Later, a user can retrieve the object from the library and obtain not only a list of genes related to the disease, but also a list of treatments that may be worthy of further investigation.

Other examples will be described in detail when describing detailed operations of the various components of the back-end 130.

30

Object Library for Genomic Data

Referring now to Figures 2A and 2B, which when placed together as indicated form Figure 2, first embodiments of a hierarchy of object superclasses and subclasses that represent genomic data according to embodiments of the invention are illustrated.

It will be understood that this hierarchy of objects may be stored in the object library 136 of Figure 1. In Figure 2, consistent with conventional object-oriented terminology, each object superclass or subclass is represented by a rectangle, the top portion of which provides the superclass or subclass name, the middle portion of which provides the associated variables, and the bottom portion of which provides the methods that are associated with the superclass and/or subclass. In addition to illustrating a hierarchy of object superclasses and subclasses that represent genomic data, Figure 2 also describes the relationship between the illustrated objects according to embodiments of the present invention.

In particular, Figure 2 illustrates a genetic object superclass 202 and the following objects which are subclasses of the genetic object 202, and which are hierarchically arranged: an experiment set object 204, an experiment object 206, a gene expression object 208, a pathology object 212, a mutation object 214, a genomic DNA object 216, a structure object 218, a protein object 222, a pathway object 224, a cancer object 226, a neurodegenerative disorder object 228, an express sequence tag (EST) object 232, a gene object 234, a stat object 236, a receptor object 238, an ion gated receptor object 242 and a G protein-coupled receptor object 244. In addition, an object builder object 252, a result set object 254, a tool interface object 256, a pattern finder object 258, a diagnosis engine object 262 and an experiment set builder object 264 also are provided.

Examples of hierarchies of object superclasses and subclasses that represent genomic data, each including associated variables for the genomic data and methods that are performed on the genomic data, as illustrated in Figure 2, now will be described. In a first example, the genetic object superclass 202 includes at least one of the experiment set subclass 204, the experiment subclass 206, the gene expression subclass 208, the pathology subclass 212, the mutation subclass 214, the genomic

subclass 216 which is a subclass of the genetic object superclass 202. In still another example, the ion gated receptor subclass 242 is a subclass of the receptor subclass 238, which is a subclass of the protein subclass 222, which is a subclass of the genetic object superclass 202. Thus, the object hierarchy illustrated in Figure 2 mimics a biological hierarchy.

A detailed explanation of some of the objects of Figure 2 and their associated methods is provided in Table 1:

Table 1

10	ExperimentSet (immutable), 204 Encapsulates the concept of a group of experiments. <i>getNumber()</i> : Returns the number of experiments in this set. <i>getExperiment(x)</i> : Returns experiment number x. 15 <i>getAllExperiments()</i> : Returns a group of experiment objects.
	Experiment (immutable), 206 Represents a microarray experiment.
20	<i>GetName()</i> : Returns the experiment name <i>GetNumber()</i> : Returns the number of GeneExpressions in the experiment <i>GetPoint(x)</i> : Returns point number x. <i>GetAllPoints()</i> : Returns a group of GeneExpression objects.
25	GeneExpression (immutable), 208 Represents a single point on a microarray slide. <i>getAccession()</i> : Returns the Accession number of the referenced gene <i>getName()</i> : Returns the name of the gene 30 <i>getExpression()</i> : Returns the Expression Level of the gene <i>getReference()</i> : Returns the reference (background level) for this gene <i>getError()</i> : Returns the measurement error in the expression level
35	Gene (immutable), 234 Represents an actual gene. <i>getAccession()</i> : Returns the Accession number of this Gene <i>getDescription()</i> : Returns the Description of this Gene <i>getOrganism()</i> : Returns the organism from which this gene was cloned 40 <i>getSequence()</i> : Returns the Gene Sequence <i>getHomology(anotherGene)</i> : Returns the percent homology between two genes <i>getProtein()</i> : Returns the protein object that this gene codes for. <i>getRelatedPathways()</i> : Returns a list of pathways in which this gene participates
45	Protein (mutable), 222 Represents a Protein.

- getAccession()*: Returns the Accession number of this protein
- getOrganism()*: Returns the organism from which this protein was cloned
- getSequence()*: Returns the protein sequence
- 5 *binds(GeneticObject)*: Returns the binding location between this protein and another genetic entity if known
- getHomology(anotherProtein)*: Returns the percent homology between two proteins.
- getRelatedPathways()*: Returns a list of pathways in which this protein participates
- getStructure()*: Returns the related structure object, if known.
- 10 *setStructure(Structure)*: Sets the proteins structure to a given obj
- getGene()*: Returns the gene object which codes for this protein.

Pathology (mutable), 212

- getName()*: Returns the name of this pathology
- 15 *getDescription()*: Returns a description of this pathology
- getTreatments()*: Returns information about possible treatments
- addTreatment(Treatment)*: Adds this treatment to the list of possible treatments
- removeTreatment(int)*: Removes a treatment from the list of possible treatments
- getRelatedGeneExpressions()*: Returns a list of possible GeneExpression objects
- 20 related to the disease.
- addRelatedGeneExpression(GeneExpression)*: Add a related Gene Expression object.
- getRelatedMutation()*: Returns a list of possible mutations related to the disease.
- addRelatedMutation(Mutation)*: Adds a related mutation

Pathway (immutable), 224

- getStages()*: Returns the number of steps in this pathway
- getRelevantGenes()*: Returns a group of Gene Objects involved in this pathway
- getStageGene(int)*: Returns the Gene Object for the given step
- getStageProtein(int)*: Returns the Protein Object produced at the given step
- 30 *getFinalProtein()*: Returns the final Protein produced by this pathway
- getProteinWithExp(Experiment)*: Returns the Protein that will be produced given the expression levels in a provided experiment

Structure (immutable), 218

- 35 *getProtein()*: Returns the protein object for which this structure was produced
- getBaseLocation(int)*: Returns the 3-d coordinates for the base at the given linear location

GeneticObject (superclass), 202

- 40 *getID()*: Returns the identification tag for the object, usually just the name.
- getSearchFields()*: Returns any information about the object that should be searchable in the database
- getType()*: Returns the object type.

ObjectBuilder (singleton), 252

createObjects(rawdata): *Creates GeneticObjects from Raw Data*

ResultSet (immutable), 254

- getResultType()*: Returns the type of result
- 50 *getResult()*: Returns a group of GeneticObjects which are the result of an operation

ToolInterface, 256

run(request, GeneticObjects): Delegates the requested task and returns a ResultSet

5 Figure 3 illustrates yet other examples of a hierarchy of object superclasses and subclasses that represent genomic data, each including associated variables for the genomic data and methods that are performed on the genomic data, according to embodiments of the invention. As shown in Figure 3, a Gene superclass includes the variable of expressing or not expressing and a gene subclass G' includes the variable of expressing under condition A but not under condition B. A gene subclass G'' is a
10 subclass of gene subclass G' and includes the variable of expressing under Protein X. A gene subclass G''' of gene subclass G' includes the property of expressing under protein Y. A further gene subclass G_a''' expresses under condition A, does not express under condition B, requires Protein Y to express and uses a sequence a. A
15 second subclass G_b''' of the subclass G''' expresses under condition A, does not express under condition B, requires Protein Y to express and uses a sequence b. Other subclasses that are not labeled in Figure 3 also may be provided.

 Thus, Figure 3 is another example of inheritance of a hierarchy of object superclasses and subclasses that represent genomic data. The particular example
20 provides a gene superclass that represents a plurality of genes, a first subclass of the gene superclass that represents genes that express under predetermined conditions, a second subclass of the first subclass that represents genes that express under the predetermined conditions and use a predetermined protein to express and a third
 subclass of the second subclass that represents genes that express under the
25 predetermined conditions, that use the predetermined protein to express and that use a predetermined sequence.

Pattern Finding

 Detailed embodiments of systems, methods and computer program products
30 for analyzing gene expression data for a plurality of genes and for a plurality of experiments, according to embodiments of the present invention, now will be described. It will be understood that these gene expression analyzing systems, methods and computer program products may be included as part of object-oriented systems, methods and computer program products for processing genomic data, for
35 example as part of the pattern finder 152 of Figure 1 and the pattern finder object 258

of Figure 2. However, it also will be understood by those having skill in the art that embodiments of systems, methods and computer program products for analyzing gene expression data for a plurality of genes and for a plurality of experiments may provide analysis that can be completed in orders of magnitude less time compared to
5 conventional self-organizing maps and may therefore be used independent of object-oriented systems, methods and computer program products for processing genomic data.

As was described above, a self-organizing map may perform a large number I (for example, between 20,000 and 50,000) of iterations of its algorithm. Each
10 iteration through the algorithm may iterate over each gene G . For each gene G , it may iterate over each node N . In order to calculate how far to move the node, it may need to perform a distance calculation in every dimension E , for example, for every experiment. Thus, the overall running time for a self-organizing map generally is proportional to the product $G \times E \times I \times N$. Moreover, since the number of different
15 patterns, and thus the number of different nodes that may be used, generally depends on the number of experiments, then the overall running time may be proportional to $G \times E \times I \times c^E$, where c is a constant describing the number of new possibilities a new experiment would bring into the system. In a particular example, assuming $N = 4$ and $I = 20,000$, a self-organizing map may need to run 3.2 million iterations. Moreover,
20 self-organizing maps generally scale exponentially as the number of new possibilities increases.

In sharp contrast, according to embodiments of the present invention, gene expression data is analyzed for a plurality of genes and for a plurality of experiments by slicing the gene expression data for the plurality of genes and for the plurality of
25 experiments into a plurality of slices that define ranges of gene expression changes for the plurality of genes among the plurality of experiments. For each gene, a determination is made as to which of the slices each of the plurality of experiments belongs.

Thus, for example, assume for the sake of simplicity that there are two
30 experiments that define an X-Y plane, wherein one of the experiments is represented on the X axis and the other experiment is represented on the Y axis. An array of points in the X-Y plane has an x,y coordinate that corresponds to a gene expression level in the two different experiments. In order to efficiently detect patterns, according to embodiments of the invention, the plane is divided into "slices", wherein

the width of these slices is defined as a percentage tolerance of the values. Figure 4 graphically illustrates division of the plane into three slices which define genes whose expression decreases significantly from Experiment 1 to Experiment 2, genes whose expression stays roughly the same and genes whose expression increases significantly. It will be understood that the "pie" shape is two dimensional because there are only two experiments. If there were three experiments, the shape would be an elongated pyramid. For four or more experiments, an accurate visual model may be difficult or impossible to visualize.

According to embodiments of the invention, each gene is iterated once to determine which section it falls into. If this pattern has not already been found, it is added to a table, along with the magnitude of difference in the gene expression values. If a pattern has been found, the gene expression values are averaged in the table. It will be understood that the term "table" is used herein to represent any indexed storage structure including linked lists, associative memory structures and other listings. Accordingly, each gene needs to be iterated only once, and assigned to a pattern grouping. For each gene, an iteration may be performed over the expression level in each experiment. Thus, the running time is proportional to the product of the number of genes G and the number of experiments. Using the example described above for self-organizing maps, only forty iterations may need to be run compared to 3.2 million iterations for a self-organizing map, as discussed above.

Figure 5 is a flowchart that illustrates embodiments of pattern finding systems, methods and/or computer program products according to embodiments of the invention. Referring now to Figure 5, the user is queried for a percentage tolerance and the experiments to be used at Block 502. This query may be embodied by presenting to the user a user interface such as shown in Figure 6. This interface may be presented, for example, at the client computer 114 using the Java Applets 116. As shown in the left portion of the user interface of Figure 6, the user may choose the experiments to analyze. As shown in the middle portion of Figure 6, the user can select "Find Expression Patterns". As shown at the right portion of Figure 6, the user may select to eliminate those genes which do not exceed the background by a given percentage. Also, a percentage tolerance is selected for grouping the genes. As shown in Figure 6, the user has selected expression levels which differ by 20%. This percentage is used to slice the gene expression data into a plurality of slices that

define ranges of gene expression changes of a plurality of genes among the plurality of experiments.

Returning again to Figure 5, at Block 504, the relevant expression data is read from the library, for example, using the server 120, request broker 132, object provider 134 and object library 136 of Figure 1. Figure 7 illustrates a format of expression data that may be read from the library at Block 504. At Block 506, an iteration is performed for each gene. At Block 508, an iteration is performed for each gene for each experiment. At Block 512, for each experiment, the gene in the experiment is compared with a previous experiment. Stated differently, a difference of a gene expression level between a present experiment and a previous experiment is determined. At Block 514, this difference is recorded as a gene's "delta-value". It will be understood that the user may be allowed to select any of a number of relationships, such as a difference relative to the same gene, a difference relative to any gene in the experiment, a difference relative to the highest absolute value for a gene or a difference relative to the highest recorded value of the gene.

At Block 516, this difference is divided by the percentage tolerance and may be rounded up or down. Thus, in some embodiments, the difference of a gene expression level is divided by the percent tolerance to obtain a value. At Block 518, this value is added or appended to the gene's profile list. Figure 8 illustrates an embodiment of a gene profile list that may be used at Block 518.

Returning to Block 508, when the experiment is completed, a test is made at Block 520 as to whether this profile already is in a results table. If yes, at Block 522, this gene's delta-value is averaged with the other genes for this profile, and, if not, at Block 524 the profile is added to the results table with the gene's delta-value. Figure 9 illustrates an embodiment of a result table that may be used at Blocks 520, 522 and 524.

After each gene is iterated at Block 506, the results table is displayed at Block 526. Figure 10 illustrates a user interface that may be used to display the result table. Each sub-block in Figure 10 can represent a gene profile, with the horizontal axis representing experiments and the vertical axis representing the average percent change between experiments.

Accordingly, large number of gene profiles from large numbers of experiments may be processed efficiently.

Diagnosis Engine

Referring now to Figure 11, embodiments of a diagnosis engine (for example, 152b of Figure 1) according to embodiments of the present invention are shown. The diagnosis engine allows a user to determine a gene expression "signature" for a particular pathology or disease. In general, the user supplies files containing gene expression test results for a group of organisms, human or otherwise, with a known pathology. The system also may contain a control group of gene expression test results from healthy individuals in a format which can be compared to the user's data. Moreover, if the system's built-in library is insufficient and/or undesirable, the user may upload control group results. The test files constitute an experiment set, and the pathology is the condition name. Diagnosis engines according to embodiments of the present invention may be used to determine the effects of any particular change, such as addition of a pharmaceutical product or environmental factor, on the gene expression pattern of an organism.

Embodiments of Figure 11 analyze the gene expression results for a test group having a pathology and the gene expression test results for a control group that is free of the pathology, in order to obtain a listing of likely candidate genes and an associated confidence level. The listing of likely candidate genes then is stored in an object-oriented library having a pathology object class, as an instance of a subclass of the pathology object class for the pathology. Thus, referring back to Figure 2, the listing of likely candidate genes may be stored as an instance of a subclass of the cancer subclass 226 or neurodegenerative disorder subclass 228 that are both subclasses of the Pathology subclass. Moreover, if the pathology subclass does not exist, a new one may be created. Accordingly, new pathologies may be added to the object library efficiently, due to the object-oriented nature of the library.

Referring now to Figure 11, at Block 1102 the user is asked for the test group and control group experiments. At Block 1104, the user is asked for the desired confidence interval. A user interface that can provide the operations of Block 1102 and 1104 is illustrated in Figure 12. As shown in Figure 12, the diagnosis engine may be selected in the middle portion of the user interface and the control group, test group and confidence level may be selected in the right portion of the user interface.

Referring back to Figure 11, at Block 1106, a new disease object is instantiated, and at Block 1108, the experiments are retrieved from the library. Iteration is performed at Block 1112 for each gene. At Block 1114 the mean and

standard deviation are calculated for the control group. At Block 1116 the mean and standard deviation for the test group are calculated. Then, at Block 1118, a t-value is calculated for the hypothesis that the means are equal. A t-test is a well known statistical test which describes whether two distributions of points are different than the statistically significant way:

$$t = \frac{\text{mean1} - \text{mean2}}{\sqrt{\text{standard error1}^2 + \text{standard error2}^2}}$$

The operations of Blocks 1114, 1116 and 1118 are well known to those having skill in the art and need not be described further herein. Moreover, it will also be understood that different statistical tests other than the t-test may be used.

At Block 1122 the confidence level is either calculated or looked up in a look-up table for this t-value. For example, if absolute value of t is less than 2.4, a 95% confidence that the values are different is obtained. Thus, if 100 trials were performed with this result, there would be a meaningful phenomenological difference in 95 of them. The formulas for calculating confidence level are well known to those having skill in the art and need not be described in detail herein.

At Block 1124 a test is made as to whether the hypothesis was rejected, meaning that the means were equal. If they were equal, then the next gene is iterated at Block 1112. If they are not equal, meaning that the hypothesis was rejected because the means are statistically different, then at Block 1126 this gene is added as a likely candidate to the disease object. When all the genes have been iterated at Block 1112, the likely candidates and confidence levels may be displayed to the user at Block 1132.

Continuing with the description of Figure 11, at Block 1134, a user may have the option to save the results. In particular, if the gene expresses in a significantly different way in a disease population and in a healthy population, it may be part of that disease's "signature", and hence may be useful as a predictor or identifier of the pathology for organisms in which the source of this function is unknown. If the user has confidence that the disease object is valid, then at Block 1136 the disease object is written to the library, for example, by adding a new instance to the cancer subclass 226, the neurodegenerative disorder subclass 228 or another subclass of the pathology

subclass 212 of Figure 2. Thus, embodiments of the invention can learn patterns that it can then apply in later analyses, to thereby extend its own functionality.

Object Generator

5 Referring now to Figure 13, embodiments of object generator systems, methods and computer program products according to the present invention (for example Block 138 of Figure 1) now will be described. Embodiments of object generators can store genomic data in an object library that includes a hierarchy of object superclasses and subclasses that represent genomic data. In general, genomic
10 data is obtained from a source that is external to the object library, wherein the genomic data includes an identification of the source. One of a plurality of object generating modules is applied to the genomic data. The object generating module is selected based on the identification of the source. The object generating module generates at least one object from the genomic data, the object including associated
15 variables for the genomic data and methods that are performed on the genomic data. The at least one object is stored in the object library.

Referring to Figure 13, the genomic data, referred to as "raw" data, is obtained from a source that is external to the object library. The genomic data includes an identification, referred to as a "tag", that identifies the source. As shown in Figure 1,
20 the raw data and tag may be obtained from the database crawler 144. However, as also shown in Figure 1, such data also may be obtained from the request broker 132, which may obtain the raw data from automated processes 146 or the client 110.

At Block 1404 the tag is looked up in a reference table of known external databases. At Block 1406 if the tag does not exist, then at Block 1408 a record is
25 made in a log that unknown data was retrieved and at Block 1412, the raw data is stored in an "unclassified data" table. However, if at Block 1406, the tag does exist, then at Block 1414 the correct filter for this tag is instantiated. As shown in Figure 1, filters 142 may include a generic array filter 142a, an NIH gene filter 142b or other filters.

30 Referring back to Figure 13, at Block 1416, the appropriate filter 142 is queried for the class of objects it will return. At Block 1418, the raw data is then passed to the object filter. For example, if the filter is an NIH object filter, Table 2 provides an example of the raw data:

Table 2

	LOCUS	T51496	338 bp	mRNA	EST	06-FEB-1995
5	DEFINITION	yb17g09.s1 Stratagene fetal spleen (#937205) Homo sapiens cDNA clone IMAGE:71488 3' similar to similar to gb:L22154 60S RIBOSOMAL PROTEIN L37A (HUMAN), mRNA sequence.				
	ACCESSION	T51496				
	VERSION	T51496.1 GI:653356				
	KEYWORDS	EST.				
10	SOURCE	human.				
	ORGANISM	Homo sapiens				
		Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Primates; Catarrhini; Hominidae; Homo.				
	REFERENCE	1 (bases 1 to 338)				
15	AUTHORS	Hillier, L., Lennon, G., Becker, M., Bonaldo, M.F., Chiapelli, B., Chissoe, S., Dietrich, N., DuBuque, T., Favello, A., Gish, W., Hawkins, M., Hultman, M., Kucaba, T., Lacy, M., Le, M., Le, N., Mardis, E., Moore, B., Morris, M., Parsons, J., Prange, C., Rifkin, L., Rohlfing, T., Schellenberg, K., Soares, M.B., Tan, F., Thierry-Mieg, J., Trevaskis, E., Underwood, K., Wohldmann, P., Waterston, R., Wilson, R. and Marra, M.				
20	TITLE	Generation and analysis of 280,000 human expressed sequence tags				
	JOURNAL	Genome Res. 6 (9), 807-828 (1996)				
	MEDLINE	97044478				
25	COMMENT	Contact: Wilson RK Washington University School of Medicine 4444 Forest Park Parkway, Box 8501, St. Louis, MO 63108 Tel: 314 286 1800 Fax: 314 286 1810 Email: est@watson.wustl.edu Insert Size: 430 Source: IMAGE Consortium, LLNL This clone is available royalty-free through LLNL; contact the IMAGE Consortium (info@image.llnl.gov) for further information. Insert Length: 430 Std Error: 0.00 Seq primer: -21m13 High quality sequence stop: 398.				
	FEATURES	Location/Qualifiers				
	source	1..338				
40		/organism="Homo sapiens"				
		/db_xref="GDB:493153"				
		/db_xref="taxon:9606"				
		/clone="IMAGE:71488"				
45		/clone_lib="Stratagene fetal spleen (#937205)"				
		/tissue_type="fetal spleen"				
		/dev_stage="fetal"				
		/lab_host="SOLR cells (kanamycin resistant)"				
50		/note="Organ: spleen; Vector: pBluescript SK-; Site_1: EcoRI; Site_2: XhoI; Cloned unidirectionally. Primer: Oligo dT. Pooled spleens. Average insert size: 1.0 kb; Uni-ZAP XR Vector; -5' adaptor sequence: 5' GAATTCGGCACGAG 3' -3' adaptor sequence: 5' CTCGAGT TTT TTT TTT TTT TTT TTT TTT 3'"				
	BASE COUNT	73 a	88 c	79 g	98 t	
	ORIGIN	1 tgttacataa attaacccat ttattatagg ccagtgatgt ctcaaagagt agaggagcgt				
55		61 ctactggtct ttcaactcct tcagtcttct gatggcggac ttaccgtga cagcggaagt				
		121 ggtattgtac gtccaggcac cgcagccact gtcttcacgc aggaaccaca gtgccagatc				
		181 cccacagctc gtctcttcat cttggttttg ccacagaaag agcaagtgt cttggcgctgc				
		241 tggctgattt caattttctt caccatttct cggaggaggc ccatagcgg gtcccgtatt				
60		301 taccgacgat cccgacttct ttgggtacgt ttggccat				
	//					

Block 1420 describes general operations that may be performed by the filters 142 (Figure 1). Specific operations will depend on the actual configuration of the data. Thus, at Block 1422, each record in the raw data is iterated. At Block 1426 a new object is instantiated. At Block 1428 the record is parsed to determine relevant information for that object. At Block 1432 the new object's properties are set

according to this relevant information. Table 3 illustrates behavior of the resulting gene object that is produced by an NIH gene filter object 142b:

Table 3

- 5 gene.getAccession()
 T51496
- 10 gene.getSequence()
 tggtacataaattaaccattttattataggccagtgatgtctcaaagagtagaggagcgtctactggtc
 tttcaactcctcagtccttctgatggcggactttaccgtgacagcgggaagtgggtattgtacgtccaggca
 ccgcagccactgtcttcatgcaggaaccacagtgccagatccccacagtcgtctcttcatcttggttt
 tgccacagaaagagcaagtgtacttggcgtgctggctgatttcaattttcttcaccattttccggagga
 ggccccatagcgggtcccgtatttaccgacgatcccgactttcttgggtacgtttggccat
- 15 gene.getOrganism()
 Homo sapiens
 Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
 Mammalia; Eutheria; Primates; Catarrhini; Hominidae; Homo.
- 20 gene.getDescription()
 yb17g09.s1 Stratagene fetal spleen (#937205) Homo sapiens cDNA clone
 IMAGE:71488 3' similar to similar to gb:L22154 60S RIBOSOMAL PROTEIN
 L37A (HUMAN), mRNA sequence.
- 25 gene.getRelatedProtein()
 This method returns the Protein Object that relates to this Gene.
- 30 gene.getBaseCount()
 342
- 35 gene.getIdentity(anotherGene)
 0.45
 Returns the percent similarity between this gene and another gene.
- 40 Returning to Block 1422, when all records have been iterated, then the group
 of new records is returned to the object generator 138 (Figure 1) at Block 1424. At
 Block 1440 the object generator 138 iterates for each object that is returned. At Block
 1442 the object is queried for its name. At Block 1444 the object is serialized into a
 bytestream, and at Block 1446, a new row is created in the object database 136
 (Figure 1) containing the name in the bytestream. Accordingly, raw data from
 external databases can be integrated automatically into an object library, so that the
 object library can add new objects based on user requests and/or independent of user
 requests.

Automated Processes

Referring now to Figure 14, operations that may be performed by automated processes (for example, Block 146 of Figure 1) according to embodiments of the invention will be described. As was described above, these automated processes may
5 be threads that run in the background and that can continuously add data to the object library 136 in a manner that can be used by other blocks of Figure 1. Accordingly, the automated processes 146 of Figure 14 can store genomic data in an object library that includes a hierarchy of object superclasses and subclasses that represent genomic data. More specifically, a new genomic data set is obtained, including a plurality of
10 records, from a source that is external to the object library. A new software object is created for each record in the new genomic data set that has no counterpart software object in an earlier version of the genomic data set that is stored in the object library. The counterpart software object and the earlier version of the genomic data set that is stored in the library is modified if the counterpart object exists in the earlier version of
15 the genomic data set. The new software object or the counterpart software object that is modified is stored in the object library. The hierarchical object model thereby may be created and/or modified in a data driven manner.

In particular, referring to Figure 14, at Block 1502, an outside database is downloaded, for example, via a network transfer, directly from physical media and/or
20 via other techniques. At Block 1504, the new data set is compared to the previous version of the data set, and at Block 1506, if nothing has changed, no further processing is performed.

However, at Block 1506, if any of the data in the data set has changed, then an iteration is performed for each record that differs at Block 1508. In particular, at
25 Block 1512 the records that differ are parsed to determine their object type. At Block 1514 if a new record is present, then a new software object is created at Block 1516. Thus, for example, a new software object for a gene expression or experiment may be created. If a new record is not present at Block 1514, then the record is parsed to determine the appropriate record to modify at Block 1518. At Block 1522 the
30 appropriate object is deserialized from the object library 136 (Figure 1) and at Block 1524 the object library is queried for the method and property list for the object type. At Block 1526 the record data is parsed to modify or create appropriate methods and properties. At Block 1528 the record data is parsed to attach return values to appropriate methods and properties. If all the data is not used, at Block 1532, an

exception is reported at Block 1534; whereas if all the data is used, then the object is again serialized at Block 1536 and placed back into the object library. Thus, the next time this object is queried, the new data is used rather than the old data. A report may be made at Block 1542 alerting the user that a new data has been incorporated. Thus, 5 objects in the object library 136 may be added and/or modified in a data-driven manner.

Database Crawler

Referring now to Figure 15, operations that may be performed by 10 embodiments of database crawlers (for example, Block 144 of Figure 1) now will be described. In general, embodiments of a database crawler can be used to create new genomic data objects for a genomic object library that contains a hierarchy of object superclasses and subclasses that represent genomic data, each including associated variables for the genomic data and methods that are performed on the genomic data. 15 Embodiments of the database crawler provide a list of genomic databases that are external of the library. A first database in the list is reviewed to determine if new information has been added thereto. The new information is obtained if the new information has been added to the first database in the list of genomic databases. The new information is converted into an instance of one of the object superclasses or 20 subclasses and is stored in the object library. The remaining databases in the list of genomic databases also are periodically reviewed, information obtained, converted and stored.

The database crawler may be used under many circumstances. For example, referring again to Figure 1, the request broker 132 can request objects from the object 25 provider 134. The object provider 134 may send back an exception if the requested object does not exist in the object library 136. However, the request broker 132 may include in the request a flag that indicates that it is willing to wait a certain amount of time in order to receive the object.

If this is the case, the object provider 134 can induce the database crawler 144 30 to spawn another process. This process can find the data that can be used to create the requested object. The database crawler 144 may have many clues as to where to find this information, such as the object type and the name of the object. For example, if the object requested is a gene, the database crawler might search the NCBI gene database. As another example, if the name of the object is known, it may be possible

for the crawler to request a single record rather than to do a batch download of many records. If the database crawler finds the record, it generates an object from it and saves the object to the object database 136.

5 Meanwhile, the object provider 134 can wait the prescribed delay time, while periodically checking the database for the new object. If the object arrives, the object provider returns the object to the request broker 132. If the time expires, it can return an exception and can tell the database crawler 144 to cease its search.

10 In particular, referring to Figure 15, the database crawler may operate in a continuous loop at Block 1602. The database list is reviewed at Block 1604 and a loop is performed for every database on the list at Block 1606. The database is reviewed, and at Block 1608, a test is made as to whether there is a new addition to the list. If yes, then the database is connected to, at Block 1616, and all of the records are downloaded. If not, then the refresh period and last refresh of the database are reviewed at Block 1612. If the refresh period has expired at Block 1624, then the
15 database is connected to and the new records are downloaded at Block 1614. Once the records have been obtained at Block 1622, the object generator 138 of Figure 1 is called with the flagged data and the database name, and an object is stored in the object library 136.

20 Once every database on the list has been probed at Block 1606, a pause is performed for a preset time and the operations begin again at Block 1602.

Experiment Set Factory

Referring now to Figure 16, embodiments of an experiment set factory (for example, Block 152c of Figure 1) now will be described. The experiment set factory
25 can start with a list of experiments and generate an experiment set therefrom.

Specifically, referring to Figure 16, at Block 1702, operations begin with a list of experiments, a minimum difference and a minimum expression. At Block 1704 experiments are retrieved from the library. At Block 1706 a new experiment set is created. Each gene is iterated at Block 1708.

30 In particular, at Block 1712, a test is made as to whether the gene differs enough across experiments, and at Block 1714 a test is made as to whether the gene is expressed highly enough in some experiments. If yes for both tests, then this gene and all its expression values are added to the experiment set. When all genes have

been iterated at Block 1708, a new experiment set is returned at Block 1718. Thus, a new experiment set may be generated from a list of experiments.

Gene Family View

5 The gene family view may be created, for example, using the Java applets 116 and servlets 124 of Figure 1 and may be displayed on the client computer 114 of Figure 1. In general, gene expression data is displayed by sorting sets of gene expression data for a plurality of experiments based on a gene family to which each gene belongs. The gene expression data then is displayed for the plurality of
10 experiments, grouped by the gene family.

 Sorting generally may be performed by obtaining a set of gene expression experimental results and a minimum change between gene expressions to be displayed. A table is created that is indexed by gene family. A family name is obtained for a first gene that changes more than the minimum change across the set of
15 gene expression results. An identification of the first gene and the gene expression level for the first gene that changes more than the minimum change across the set of gene expression results is stored, under the index of the family name that is obtained. Obtaining a family name and storing is repeatedly performed for remaining genes in the set of gene expression results that change by more than the minimum change, to
20 thereby store in the table the gene expression results, grouped by family name.

 In embodiments of the invention, displaying the gene expression data for the plurality of experiments, grouped by the gene family, may be performed by displaying an array of boxes, each of which includes a gene family name associated therewith. An array of subboxes is displayed in each box, wherein each subbox corresponds to a
25 gene in the associated gene family. An indicium is displayed in each subbox, that identifies the associated expression level for the associated gene. The indicium may be a brightness level that is in direct proportion to, for example, logarithmic to, the associated expression level for the associated gene.

 More specifically, referring to Figure 17, at Block 1802, a user is queried for
30 the minimum change between genes, for example using an appropriate user interface. At Block 1804, the user is queried as to which experiments to view using an appropriate user interface. It will be understood that the user interface of Figure 18 may be used. As shown in Figure 18, the experiments may be selected at the left portion of the user interface.

Returning again to Figure 17, at Block 1806, the experiments are retrieved from the object library 136. At Block 1808, a new table, referred to as a "hash table" (a programming term for indexed table), is created that is indexed by gene family. Then, at Block 1812, operations are iterated for each gene, to eliminate those genes
5 that do not have sufficient expression change to be meaningful. In particular, at Block 1814, a test is made as to whether the gene's expression changes enough across each experiment. If yes, then the gene is queried for its family name at Block 1816. If there is a key in the hash table for this family at Block 1818, then the gene is added to the appropriate place in the hash table at Block 1822. If not, at Block 1824 the family
10 is added as a key to the hash table, and the gene is added to the newly added family.

After the genes that do not change enough across each experiment have been eliminated by performing the iterations of Block 1812, then a view for display is created by performing the iterations of Block 1826 for each key in the hash table that corresponds to a family. In particular, for each key of the hash table at Block 1828,
15 the key name is drawn with a large box. Then, at Block 1832, for each gene under this key, the gene is queried for its expression level at Block 1834. At Block 1836 a small box, referred to as a subbox, is drawn within the large box whose brightness is determined by the expression level. Figure 19 illustrates an example of a gene family view display. As shown in Figure 19, one or more arrays of boxes 1910 are
20 displayed, each including an array of subboxes 1920 that include a color and/or intensity indicium 1930. The gene identifications may occur adjacent each box and/or in a separate section of the display.

Gene expression data thus may be displayed in a useful manner. It will be understood that the gene family view may be provided independent of object-oriented
25 genomic data processing systems, methods and computer program products.

Scatter Plot With Error Cloud

Embodiments of the present invention also may display gene expression data for two or more experiments in a manner which can indicate the errors in the gene
30 expression as well as the gene expression data itself. This display may be used independent of object-oriented genomic data processing systems, methods and computer program products.

In particular, gene expressions for first and second experiments may be displayed by displaying a plurality of points in a two-dimensional coordinate system,

each point being located at coordinates that correspond to the gene expression data for a gene in the first and second experiments. An error cloud is displayed around at least some of the points, indicating an error in the gene expression data for the associated gene, in the first and second experiments. More specifically, the plurality of points
5 may be displayed by displaying an X-Y coordinate system. For each gene whose expression level changes by more than a predetermined amount, a y coordinate is assigned as the gene's expression in the first experiment and an x coordinate is assigned as the gene's expression in the second experiment. For each of the genes, a point is displayed that corresponds to the x,y coordinates on the X-Y coordinate
10 system, for each of the genes.

The error cloud may be displayed by assigning YE as the gene's error in the first experiment and XE as the gene's error in the second experiment. A cloud is then displayed that is centered at x,y and having a height YE and a width XE on the X-Y coordinate system. The cloud preferably is lighter than the point.

15 Figure 21 is a user interface of a scatter plot with an error cloud. Two experiments are shown. The first, HL60(t0), is plotted on the Y axis and the second, HL60(t4), is plotted on the X-axis. The expression level of the gene in one experiment thereby may be compared to that of the same gene in another experiment by judging the distance from the line $Y=X$, as is conventionally done. However,
20 Figure 21 also illustrates the addition of larger, preferably dimmer, ellipses or clouds around the points, which can give the viewer an idea of the amount of error in each point. The width and height of the ellipse may correspond to the standard error of the gene experiments HL60(t4) and HL60(t0), respectively. Thus, a point representing a gene surrounded by a very large circle can indicate that there is a lot of error in that
25 gene's expression data in both experiments. A tall narrow ellipse suggests a poor measurement in HL60(t0) but a good measurement in HL60(t4), and vice versa, yielding a wide flat ellipse.

The error cloud may be useful because it can provide the user an immediate impression of how good the data is, without having to look in a chart of color codes as
30 with "spot brightness" display techniques. Moreover, it also can allow the user to compare two neighboring points. For example, if one point is within another point's error cloud, there is a strong possibility that the two points are in fact equal in both experiments. Moreover, a point that appears to deviate from the $Y=X$ line that has an

error cloud that reaches close to the line may indicate that the deviation may not be as interesting as was first anticipated.

Referring to Figure 20, operations for displaying a scatter plot with an error cloud are shown. A user interface may be used to obtain from the user a minimum
5 change between genes (Block 2102) and the two experiments to view (Block 2104). Then, at Block 2106, the experiments are retrieved from the database. At Block 2108 operations are performed for each gene. In particular, at Block 2112, a test is made as to whether this gene's expression changes enough across each experiment. If yes, then at Block 2114, Y is set to the gene's expression in the first experiment, and at
10 Block 2116 X is set to the gene's expression in the second experiment. At Block 2118 YE is set to the gene's error in the first experiment, and at Block 2122 XE is set to the gene's error in the second experiment. At Block 2124, a light-colored circle is drawn centered around X,Y with height YE and width XE, and at Block 2126 a dark-colored circle is drawn centered at X,Y, preferably of minimal visible size. It also will be
15 understood that a three-dimensional error cloud also may be generated and displayed, corresponding to three experiments. Higher order error clouds also may be generated, but they may be difficult to display.

Conclusion

20 As was described extensively herein, genomic data processing systems, methods and computer program products according to embodiments of the present invention can model genomic data the same way in a computer that it exists in a biological cell: code and data coexist in one unit. Object orientation allows the object library to mimic biological objects. Moreover, objects may be self-modifying, to
25 allow new object classes and instances to be created for extensibility. Thus, large volumes of raw data can be converted into usable, organized hierarchical structures. Moreover, the object library can learn and thereby create hypotheses without user input. The object library may be enhanced by calling external database information on a periodic basis to update the object library.

30 Scalable software models of cellular activity may be created, and the properties of these models may change (either autonomously or from user input) as new information is uncovered. Data from diverse databases may be incorporated into the object library, so that cellular function may be predicted from diverse genomic data.

Accordingly, meaningful information such as gene sequences, protein expression levels and confluence and divergence points between diverse microarrays can be extracted efficiently. Users can identify and verify obscure or nonintuitive relationships in their data sets. Thus, where existing gene expression analytical tools may only provide data visualization as a means of answering preexisting questions, 5 embodiments of the present application can help users ask the right questions of their data sets and/or identify relationships without guidance from the user. These attributes can accelerate the drug discovery process for pharmaceutical and biotechnology companies and can provide new insights into medical therapeutics.

10 In the drawings and specification, there have been disclosed typical preferred embodiments of the invention and, although specific terms are employed, they are used in a generic and descriptive sense only and not for purposes of limitation, the scope of the invention being set forth in the following claims.

What is Claimed is:

1. A method of processing genomic data comprising:
providing a hierarchy of object superclasses and subclasses that represent genomic data, each including associated variables for the genomic data and methods that are performed on the genomic data; and
5 sending messages between the hierarchy of object superclasses and subclasses that represent genomic data, to thereby store, recall and analyze the genomic data.
2. A method according to Claim 1 wherein the step of providing a hierarchy of object superclasses and subclasses that represent genomic data comprises:
providing a genetic object superclass and at least one of an experiment set
5 subclass, an experiment subclass, a gene expression subclass, a pathology subclass, a mutation subclass, a genomic DNA subclass, a structure subclass, a protein subclass and a pathway subclass of the genetic object superclass.
3. A method according to Claim 1 wherein the step of providing a hierarchy of object superclasses and subclasses that represent genomic data comprises:
providing an experiment set superclass that represents at least one microarray
5 experiment, an experiment subclass of the experiment set superclass that represents a single microarray experiment and a gene expression class of the experiment subclass that represents a single point in a microarray experiment.
4. A method according to Claim 1 wherein the step of providing a hierarchy of object superclasses and subclasses that represent genomic data comprises:
providing a gene superclass that represents at least one gene, a first subclass of
5 the gene superclass that represents genes that express under predetermined conditions, a second subclass of the first subclass that represents genes that express under the predetermined conditions and use a predetermined protein to express and a third subclass of the second subclass that represents genes that express under the predetermined conditions, that use the predetermined protein to express and that use a
10 predetermined gene sequence.

5. A method according to Claim 1 wherein the step of providing a hierarchy of object superclasses and subclasses that represent genomic data comprises:
- providing a genetic object superclass, a pathology subclass of the genetic object superclass and a cancer subclass of the pathology subclass.
6. A method according to Claim 1 wherein the step of providing a hierarchy of object superclasses and subclasses that represent genomic data comprises:
- providing a genetic object superclass, a genomic DNA subclass of the genetic object superclass, a gene subclass of the genomic DNA subclass and an expressed sequence tag subclass of the genomic DNA subclass.
7. A method according to Claim 1 wherein the step of providing a hierarchy of object superclasses and subclasses that represent genomic data comprises:
- providing a genetic object superclass, a protein subclass of the genetic object superclass, a receptor subclass of the protein subclass and an ion gated receptor subclass of the receptor subclass.
8. A method according to Claim 1 wherein the step of sending messages between the hierarchy of object superclasses and subclasses that represent genomic data comprises sending messages in response to a user request to store, recall or analyze the genomic data.
9. A method according to Claim 1 wherein the step of sending messages between the hierarchy of object superclasses and subclasses that represent genomic data comprises sending messages in response to automated processes that run independent of user requests to store, recall or analyze the genomic data.
10. A method according to Claim 1 wherein the step of sending messages between the hierarchy of object superclasses and subclasses that represent genomic data comprises sending messages in response to genomic data that is obtained by

5 crawling a network to store, recall or analyze the genomic data that is obtained by crawling the network.

11. A method according to Claim 1 wherein the step of sending messages between the hierarchy of object superclasses and subclasses that represent genomic data comprises sending messages between the hierarchy of object superclasses and subclasses that represent genomic data, to analyze the genomic data by finding
5 patterns of gene expression levels among at least one gene in at least one experiment.

12. A method according to Claim 1 wherein the step of sending messages between the hierarchy of object superclasses and subclasses that represent genomic data comprises sending messages between the hierarchy of object superclasses and subclasses that represent genomic data, to analyze a gene expression signature for a
5 pathology.

13. A method according to Claim 1 further comprising:
generating an instance of at least one of the object superclasses and subclasses that represent genomic data, in response to receipt of raw genomic data.

14. A method according to Claim 13 wherein the step of generating an instance comprises generating an instance of at least one of the object superclasses and subclasses that represent genomic data, in response to raw genomic data that is received from at least one of a user, a process that is automatically run and a network
5 crawler.

15. A method according to Claim 13 wherein the step of generating an instance comprises:
providing a plurality of filters, each of which converts raw data from a predetermined source into an instance of at least one of the object superclasses and
5 subclasses that represent genomic data; and
applying a selected one of the filters to the raw genomic data, based upon the source of the raw data.

16. A method according to Claim 11 further comprising:

displaying the patterns of gene expression levels among the at least one gene in the at least one experiment that meet a predefined percentage change in expression level between the at least one experiment.

17. A method according to Claim 11 further comprising:

displaying the patterns of gene expression levels among the plurality of genes in the at least one experiment as a function of gene families to which the at least one gene in the at least one experiment belongs.

18. A method according to Claim 11 further comprising:

displaying the patterns of gene expression levels among the at least one gene in the at least one experiment as a scatter plot of points of gene expression levels among at least one gene in at least two experiments, wherein at least some of the
5 points are surrounded by an error cloud that indicates an amount of error in the at least two experiments.

19. A system for processing genomic data comprising:

a hierarchy of object superclasses and subclasses that represent genomic data, each including associated variables for the genomic data and methods that are performed on the genomic data; and

5 an object messenger that sends messages between the hierarchy of object superclasses and subclasses that represent genomic data, to thereby store, recall and analyze the genomic data.

20. A system according to Claim 19 wherein the hierarchy of object superclasses and subclasses that represent genomic data comprises:

a genetic object superclass and at least one of an experiment set subclass, an experiment subclass, a gene expression subclass, a pathology subclass, a mutation
5 subclass, a genomic DNA subclass, a structure subclass, a protein subclass and a pathway subclass of the genetic object superclass.

21. A system according to Claim 19 wherein the hierarchy of object superclasses and subclasses that represent genomic data comprises:

an experiment set superclass that represents at least one microarray experiment, an experiment subclass of the experiment set superclass that represents a single microarray experiment and a gene expression class of the experiment subclass that represents a single point in a microarray experiment.

22. A system according to Claim 19 wherein the hierarchy of object superclasses and subclasses that represent genomic data comprises:

a gene superclass that represents at least one gene, a first subclass of the gene superclass that represents genes that express under predetermined conditions, a second subclass of the first subclass that represents genes that express under the predetermined conditions and use a predetermined protein to express and a third subclass of the second subclass that represents genes that express under the predetermined conditions, that use the predetermined protein to express and that use a predetermined gene sequence.

23. A method according to Claim 19 wherein the hierarchy of object superclasses and subclasses that represent genomic data comprises:

a genetic object superclass, a pathology subclass of the genetic object superclass and a cancer subclass of the pathology subclass.

24. A system according to Claim 19 wherein the hierarchy of object superclasses and subclasses that represent genomic data comprises:

a genetic object superclass, a genomic DNA subclass of the genetic object superclass, a gene subclass of the genomic DNA subclass and an expressed sequence tag subclass of the genomic DNA subclass.

25. A system according to Claim 19 wherein the hierarchy of object superclasses and subclasses that represent genomic data comprises:

a genetic object superclass, a protein subclass of the genetic object superclass, a receptor subclass of the protein subclass and an ion gated receptor subclass of the receptor subclass.

26. A system according to Claim 19 wherein the object messenger is configured to send messages between the hierarchy of object superclasses and

subclasses that represent genomic data in response to a user request to store, recall or analyze the genomic data.

27. A system according to Claim 19 wherein the object messenger is configured to send messages between the hierarchy of object superclasses and subclasses that represent genomic data in response to automated processes that run independent of user requests to store, recall or analyze the genomic data.

28. A system according to Claim 19 wherein the object messenger is configured to send messages between the hierarchy of object superclasses and subclasses that represent genomic data in response to genomic data that is obtained by crawling a network to store, recall or analyze the genomic data that is obtained by
5 crawling the network.

29. A system according to Claim 19 wherein the object messenger is configured to send messages between the hierarchy of object superclasses and subclasses that represent genomic data, to analyze the genomic data by finding patterns of gene expression levels among at least one gene in at least one experiment.
5

30. A system according to Claim 19 wherein the object messenger is configured to send messages between the hierarchy of object superclasses and subclasses that represent genomic data, to analyze a gene expression signature for a pathology.

31. A system according to Claim 19 further comprising:
an instance generator that is configured to generate an instance of at least one of the object superclasses and subclasses that represent genomic data, in response to receipt of raw genomic data.

32. A system according to Claim 31 wherein the instance generator is further configured to generate an instance of at least one of the object superclasses and subclasses that represent genomic data, in response to raw genomic data that is received from at least one of a user, a process that is automatically run and a network
5 crawler.

33. A method according to Claim 31 wherein the instance generator comprises:

a plurality of filters, each of which converts raw genomic data from a predetermined source into an instance of at least one of the object superclasses and subclasses that represent genomic data; and

means for applying a selected one of the filters to the raw genomic data, based upon the source of the raw data.

34. A system according to Claim 29 further comprising:

a display that is configured to display the patterns of gene expression levels among the at least one gene in the at least one experiment that meet a predefined percentage change in expression level between the at least one experiment.

35. A system according to Claim 29 further comprising:

a display that is configured to display the patterns of gene expression levels among the at least one gene in the at least one experiment as a function of gene families to which the at least one gene in the at least one experiment belongs.

36. A system according to Claim 29 further comprising:

a display that is configured to display the patterns of gene expression levels among the at least one gene in the at least one experiment as a scatter plot of points of gene expression levels among at least one gene in at least two experiments, wherein at least some of the points are surrounded by an error cloud that indicates an amount of error in the at least two experiments.

37. A computer program product that processes genomic data, the computer program product comprising a computer usable storage medium having computer-readable program code embodied in the medium, the computer-readable program code comprising:

computer-readable program code that is configured to provide a hierarchy of object superclasses and subclasses that represent genomic data, each including associated variables for the genomic data and methods that are performed on the genomic data; and

10 computer-readable program code that is configured to send messages between the hierarchy of object superclasses and subclasses that represent genomic data, to thereby store, recall and analyze the genomic data.

38. A computer program product according to Claim 37 wherein the computer-readable program code that is configured to provide a hierarchy of object superclasses and subclasses that represent genomic data comprises:

5 computer-readable program code that is configured to provide a genetic object superclass and at least one of an experiment set subclass, an experiment subclass, a gene expression subclass, a pathology subclass, a mutation subclass, a genomic DNA subclass, a structure subclass, a protein subclass and a pathway subclass of the genetic object superclass.

39. A computer program product according to Claim 37 wherein the computer-readable program code that is configured to provide a hierarchy of object superclasses and subclasses that represent genomic data comprises:

5 computer-readable program code that is configured to provide an experiment set superclass that represents at least one microarray experiment, an experiment subclass of the experiment set superclass that represents a single microarray experiment and a gene expression class of the experiment subclass that represents a single point in a microarray experiment.

40. A computer program product according to Claim 37 wherein the computer-readable program code that is configured to provide a hierarchy of object superclasses and subclasses that represent genomic data comprises:

5 computer-readable program code that is configured to provide a gene superclass that represents at least one gene, a first subclass of the gene superclass that represents genes that express under predetermined conditions, a second subclass of the first subclass that represents genes that express under the predetermined conditions and use a predetermined protein to express and a third subclass of the second subclass that represents genes that express under the predetermined
10 conditions, that use the predetermined protein to express and that use a predetermined gene sequence.

41. A computer program product according to Claim 37 wherein the computer-readable program code that is configured to provide a hierarchy of object superclasses and subclasses that represent genomic data comprises:

5 computer-readable program code that is configured to provide a genetic object superclass, a pathology subclass of the genetic object superclass and a cancer subclass of the pathology subclass.

42. A computer program product according to Claim 37 wherein the computer-readable program code that is configured to provide a hierarchy of object superclasses and subclasses that represent genomic data comprises:

5 computer-readable program code that is configured to provide a genetic object superclass, a genomic DNA subclass of the genetic object superclass, a gene subclass of the genomic DNA subclass and an expressed sequence tag subclass of the genomic DNA subclass.

43. A computer program product according to Claim 37 wherein the computer-readable program code that is configured to provide a hierarchy of object superclasses and subclasses that represent genomic data comprises:

5 computer-readable program code that is configured to provide a genetic object superclass, a protein subclass of the genetic object superclass, a receptor subclass of the protein subclass and an ion gated receptor subclass of the receptor subclass.

44. A computer program product according to Claim 37 wherein the computer-readable program code that is configured to send messages between the hierarchy of object superclasses and subclasses that represent genomic data comprises computer-readable program code that is configured to send messages in response to a
5 user request to store, recall or analyze the genomic data.

45. A computer program product according to Claim 37 wherein the computer-readable program code that is configured to send messages between the hierarchy of object superclasses and subclasses that represent genomic data comprises computer-readable program code that is configured to send messages in response to
5 automated processes that run independent of user requests to store, recall or analyze the genomic data.

46. A computer program product according to Claim 37 wherein the computer-readable program code that is configured to send messages between the hierarchy of object superclasses and subclasses that represent genomic data comprises computer-readable program code that is configured to send messages in response to
5 genomic data that is obtained by crawling a network to store, recall or analyze the genomic data that is obtained by crawling the network.

47. A computer program product according to Claim 37 wherein the computer-readable program code that is configured to send messages between the hierarchy of object superclasses and subclasses that represent genomic data comprises computer-readable program code that is configured to send messages between the
5 hierarchy of object superclasses and subclasses that represent genomic data, to analyze the genomic data by finding patterns of gene expression levels among at least one gene in at least one experiment.

48. A computer program product according to Claim 37 wherein the computer-readable program code that is configured to send messages between the hierarchy of object superclasses and subclasses that represent genomic data comprises computer-readable program code that is configured to send messages between the
5 hierarchy of object superclasses and subclasses that represent genomic data, to analyze a gene expression signature for a pathology.

49. A computer program product according to Claim 37 further comprising:

computer-readable program code that is configured to generate an instance of at least one of the object superclasses and subclasses that represent genomic data, in
5 response to receipt of raw genomic data.

50. A computer program product according to Claim 49 wherein the computer-readable program code that is configured to generate an instance comprises computer-readable program code that is configured to generate an instance of at least one of the object superclasses and subclasses that represent genomic data, in response

- 5 to raw genomic data that is received from at least one of a user, a process that is automatically run and a network crawler.

51. A computer program product according to Claim 49 wherein the computer-readable program code that is configured to generate an instance comprises:
computer-readable program code that is configured to provide a plurality of filters, each of which converts raw data from a predetermined source into an instance
5 of at least one of the object superclasses and subclasses that represent genomic data; and
computer-readable program code that is configured to apply a selected one of the filters to the raw genomic data, based upon the source of the raw data.

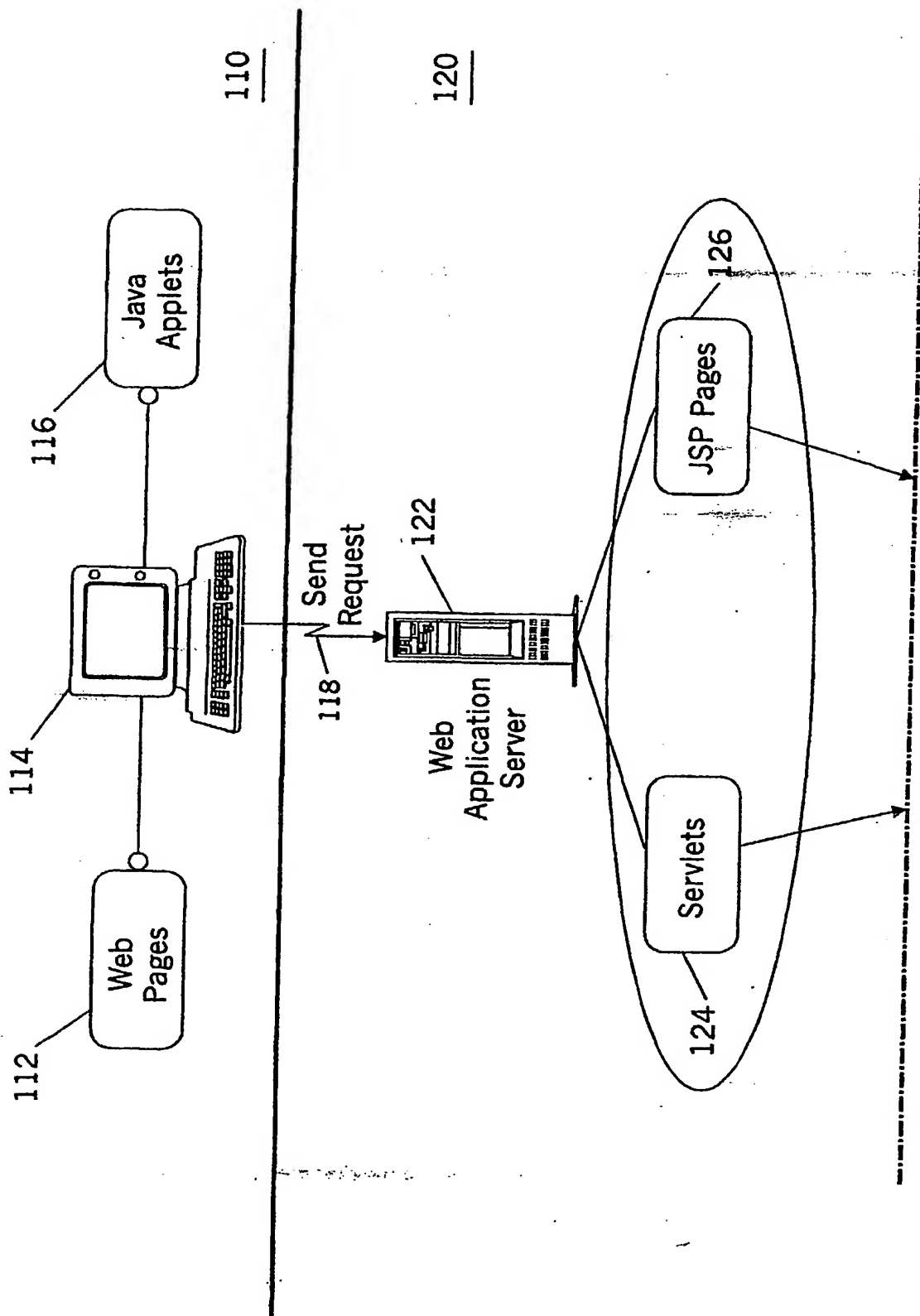
52. A computer program product according to Claim 47 further comprising:
computer-readable program code that is configured to display the patterns of gene expression levels among the at least one gene in at least one experiment that
5 meet a predefined percentage change in expression level between the at least one experiment.

53. A computer program product according to Claim 47 further comprising:
computer-readable program code that is configured to display the patterns of gene expression levels among the at least one gene in the at least one experiment as a
5 function of gene families to which the at least one gene in the at least one experiment belongs.

54. A computer program product according to Claim 47 further comprising:
computer-readable program code that is configured to display the patterns of gene expression levels among the at least one gene in the at least one experiment as a
5 scatter plot of points of gene expression levels among at least one gene in at least two experiments, wherein at least some of the points are surrounded by an error cloud that indicates an amount of error in the at least two experiments.

1/26

FIG. 1A



2/26

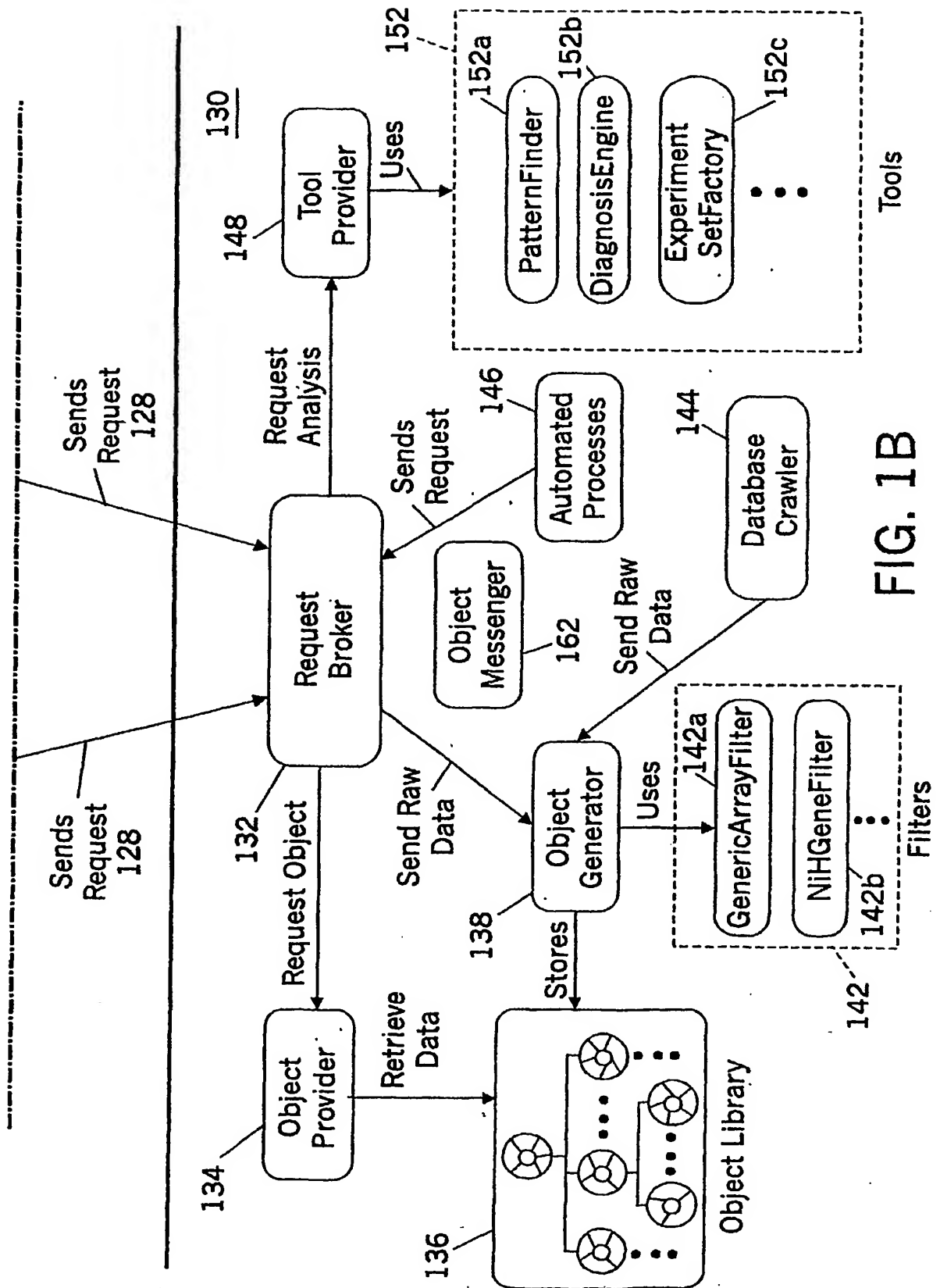


FIG. 1B

FIG. 2A
FIG. 2B
FIG. 2C

FIG. 2

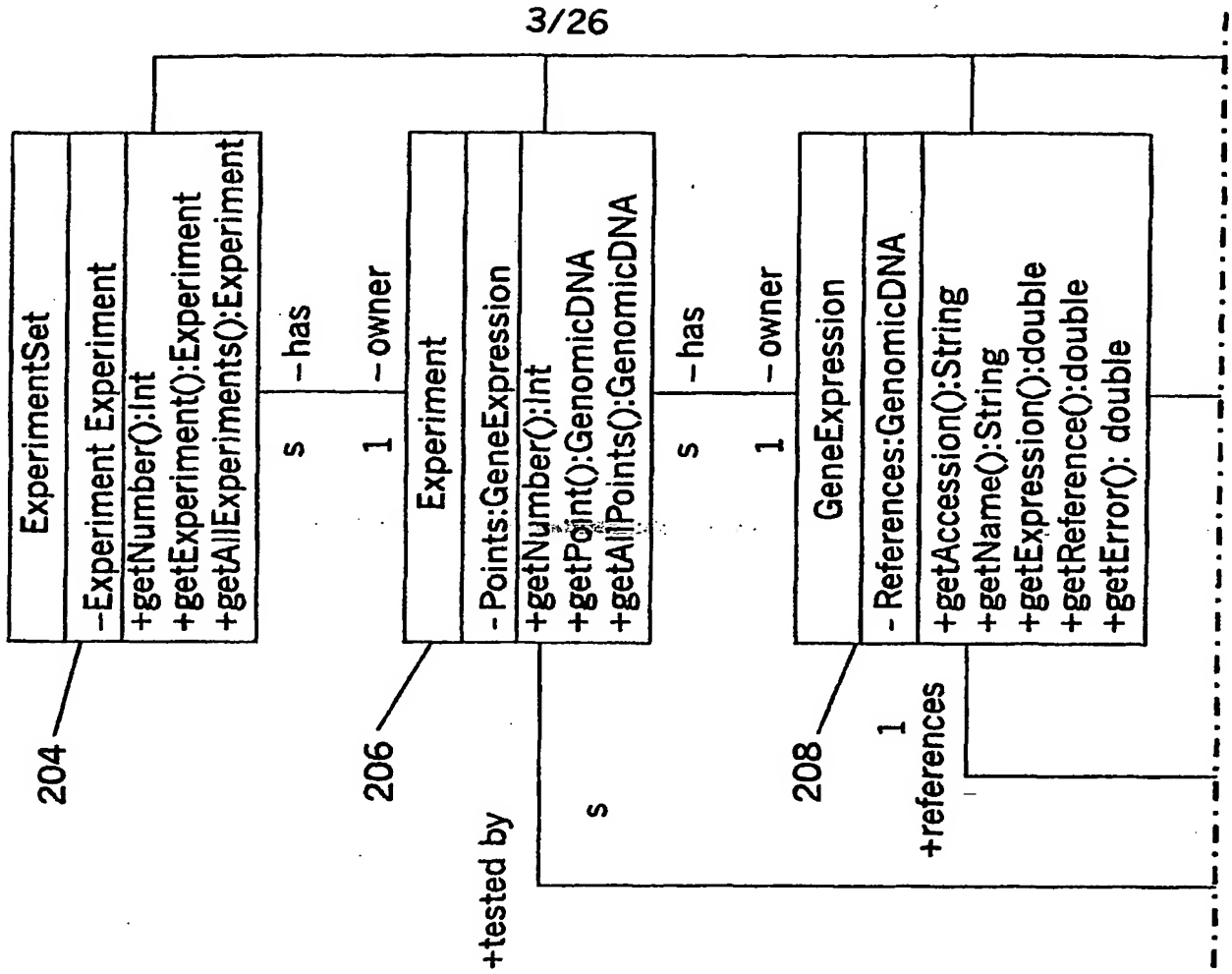
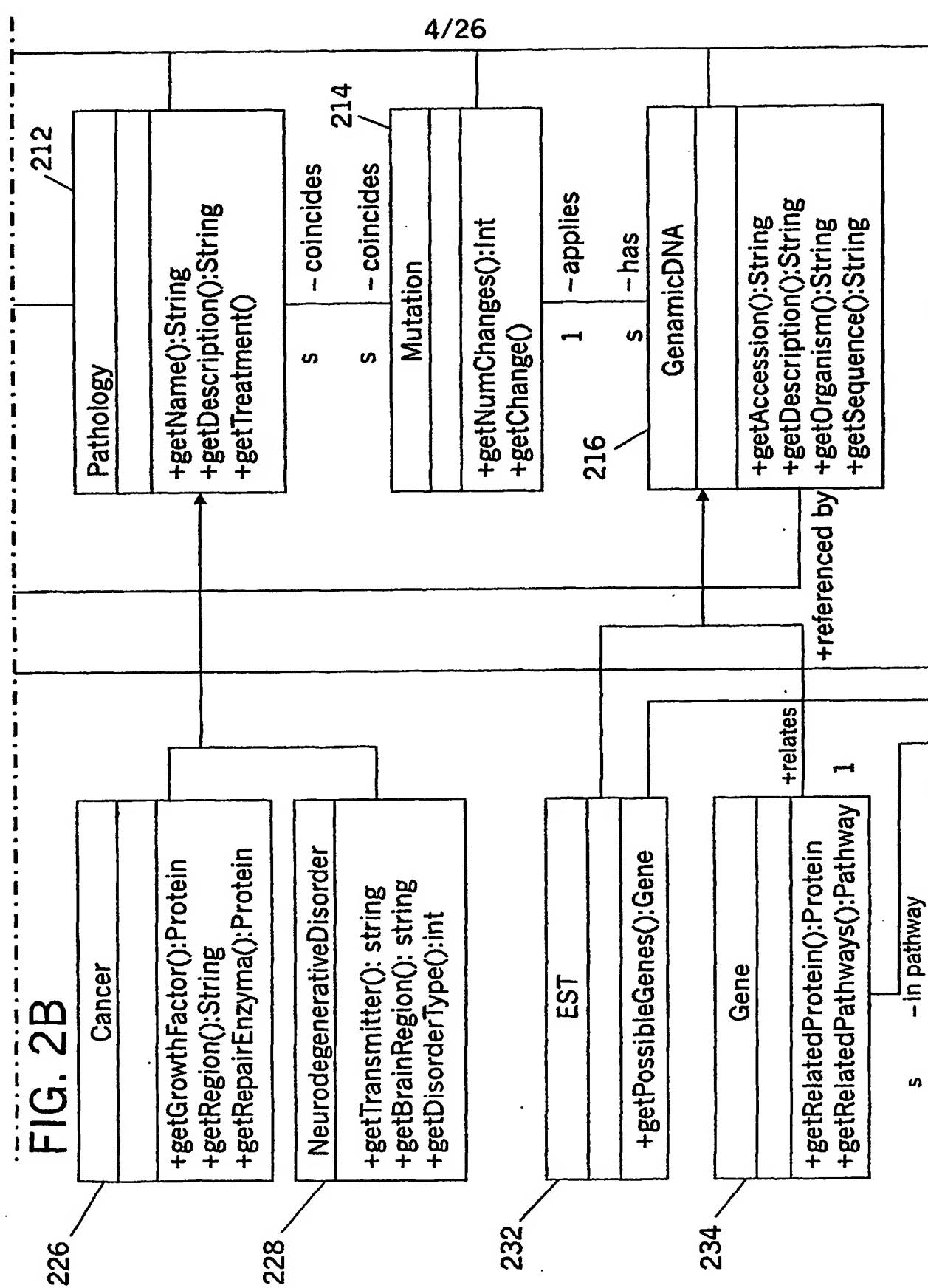


FIG. 2A



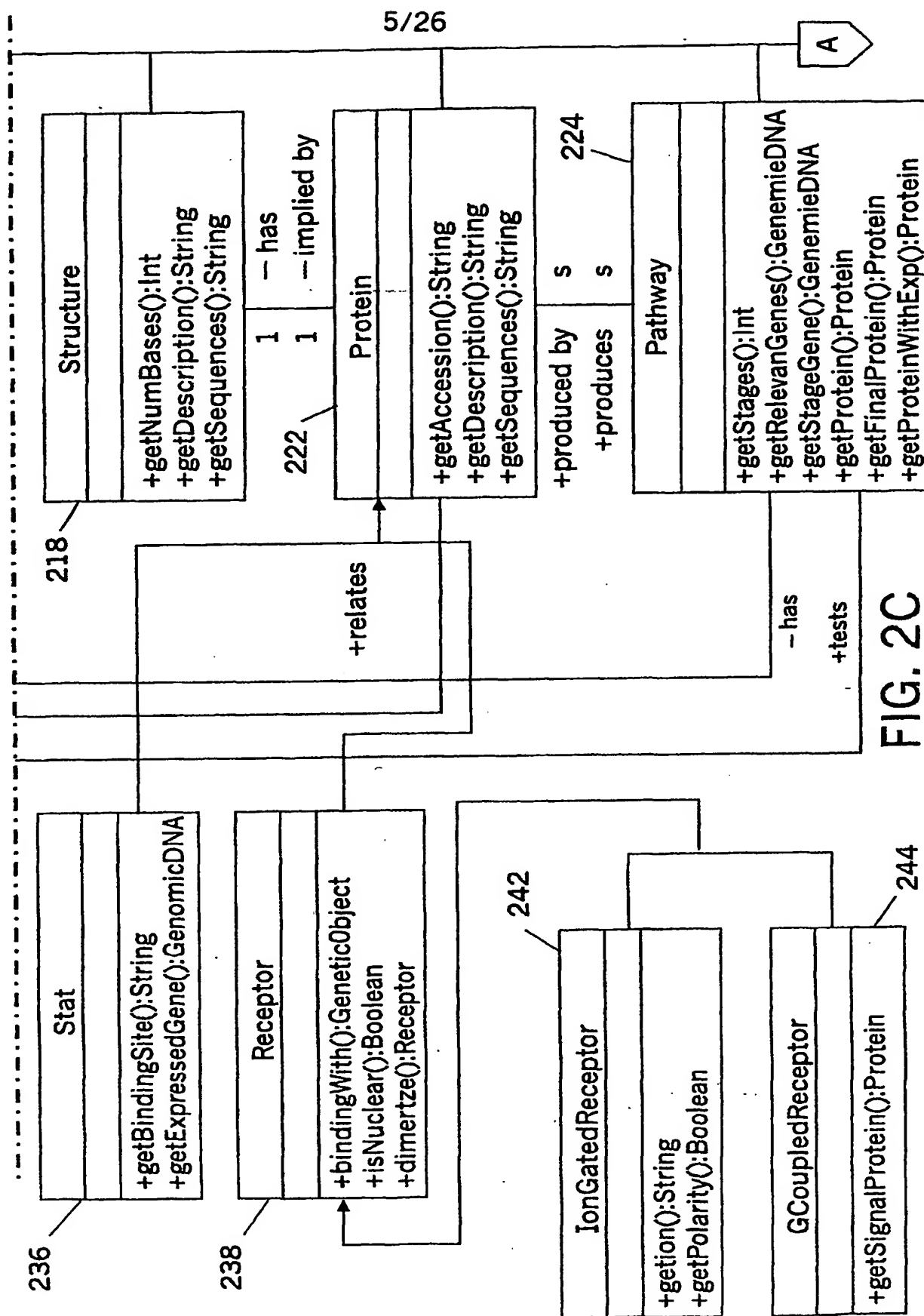
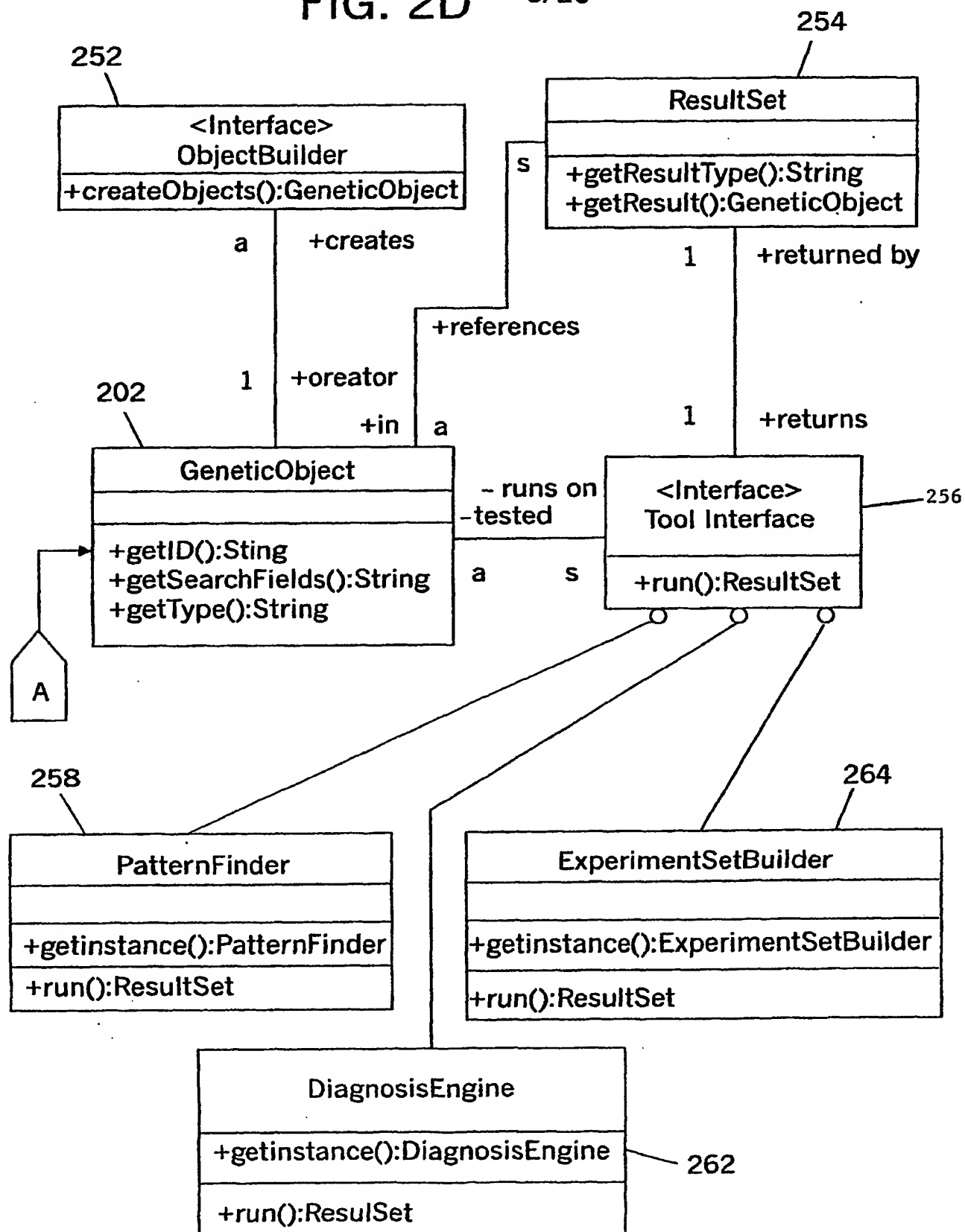


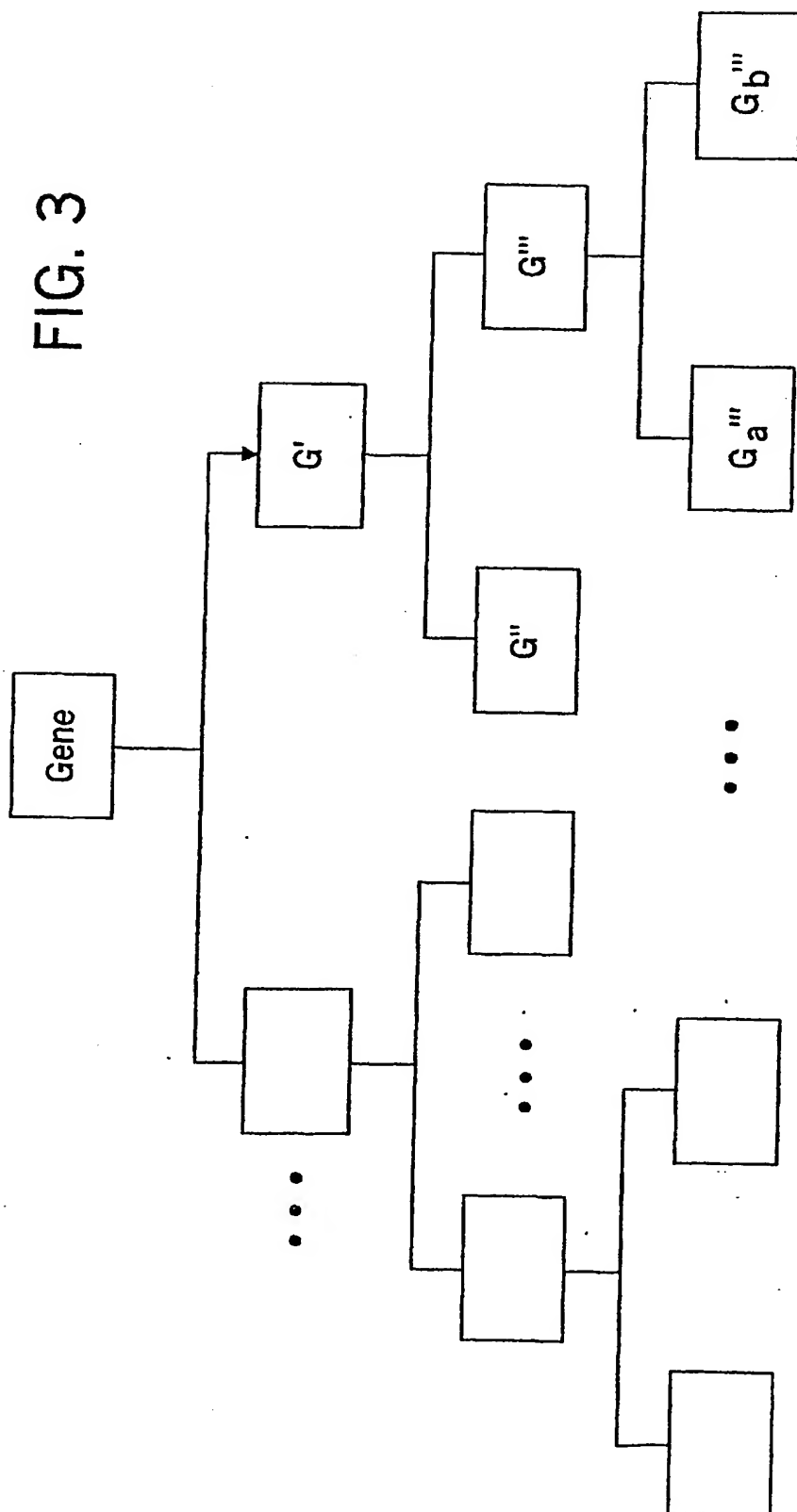
FIG. 2C

FIG. 2D 6/26



7/26

FIG. 3



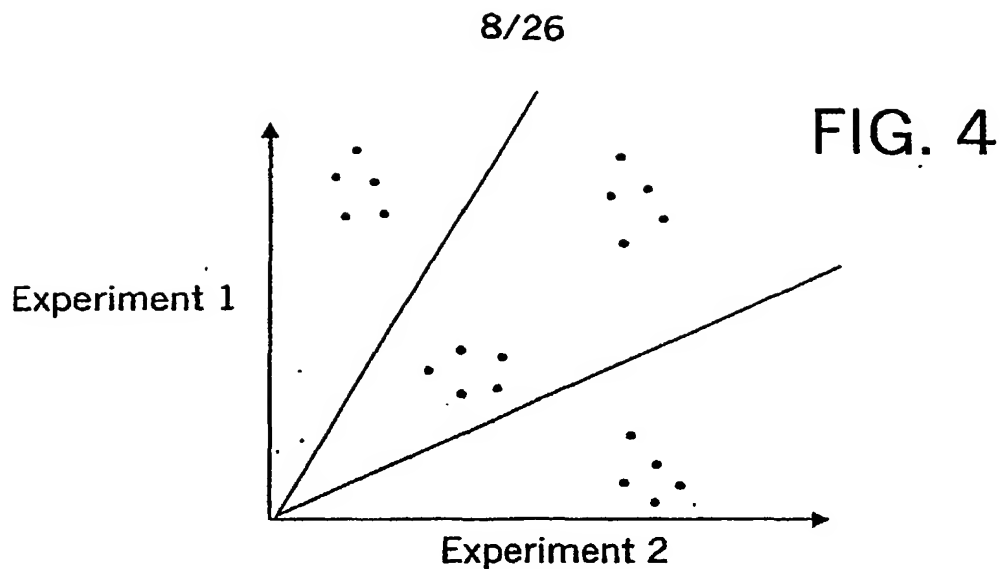


FIG. 7

Gene Name	Expression Level
<gene names>	<numbers representing expression level>
• • •	• • •

FIG. 8

	Change between experiments divided by tolerance (rounded down)
1-2	• • •
2-3	• • •
3-4	• • •
etc.	• • •

FIG. 9

Profile	Delta Values
<gene profile>	<average percent change between experiments>
• • •	• • •

9/26

FIG. 5

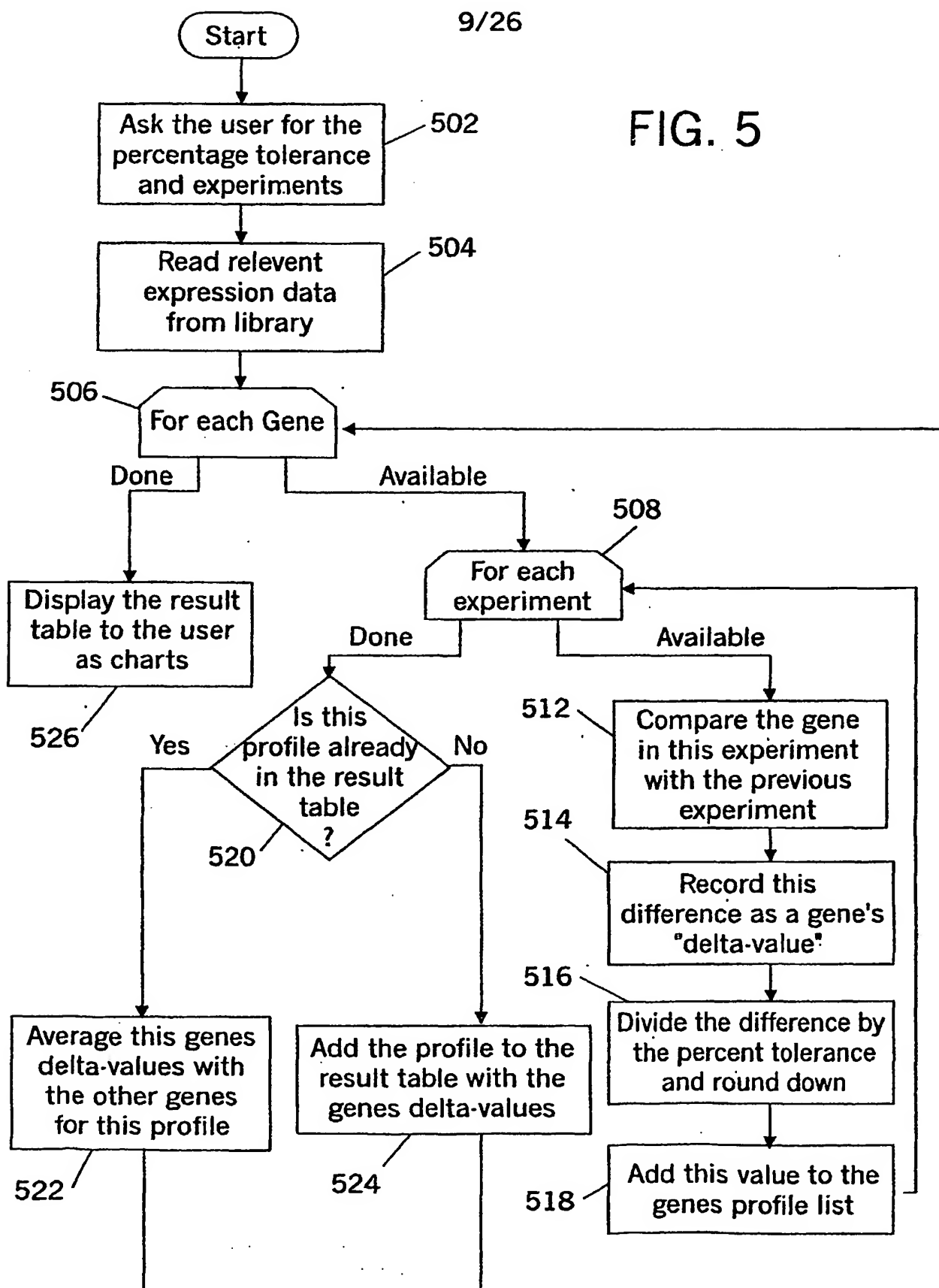


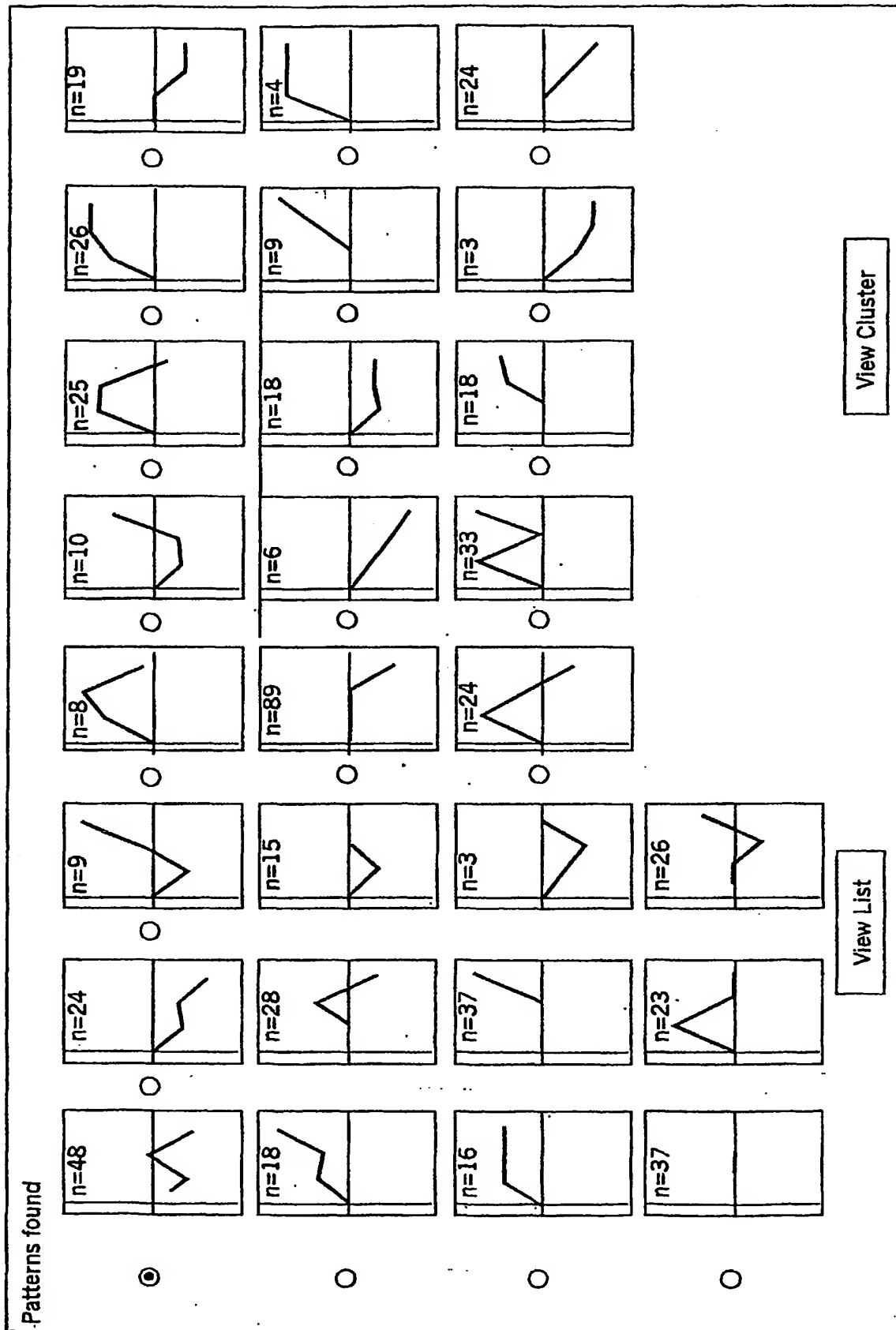
FIG. 6

10/26

Experiments	Analysis	Views
<p>Choose Experiments to Analyze</p> <div> <input type="checkbox"/> Array Experiments <input type="checkbox"/> Personal <input type="checkbox"/> Experiments </div> <div> <input checked="" type="checkbox"/> HL60 (0h) <input checked="" type="checkbox"/> HL60 (0.25h) <input checked="" type="checkbox"/> HL80 (4h) <input checked="" type="checkbox"/> HL60 (24h) <input type="checkbox"/> Sample 1 <input type="checkbox"/> Sample 2 <input type="checkbox"/> Sample 3 <input type="checkbox"/> SAMPLE5 </div> <div> <input type="checkbox"/> Disease/Condition <input type="checkbox"/> Workgroup <input type="checkbox"/> Public </div> <p>Add New Experiment</p>	<p>Choose Analysis Method</p> <p> <input type="radio"/> View all Genes <input type="radio"/> View marked Genes <input checked="" type="radio"/> Find Expression Patterns <input type="radio"/> Pathway Viewer <input type="radio"/> Diagnosis Engine </p>	<p>Find Expression Patterns</p> <p>Show only genes which exceed background by: <input type="text" value="0.0x"/></p> <p>Genes are different which differ in expression by: <input type="text" value="20%"/></p> <p>Find</p>
<p>Normalize Data</p> <p> <input type="checkbox"/> Experiment to Self <input type="checkbox"/> Genes to Background <input type="checkbox"/> Control: <input type="text"/> </p>		

11/26

FIG. 10

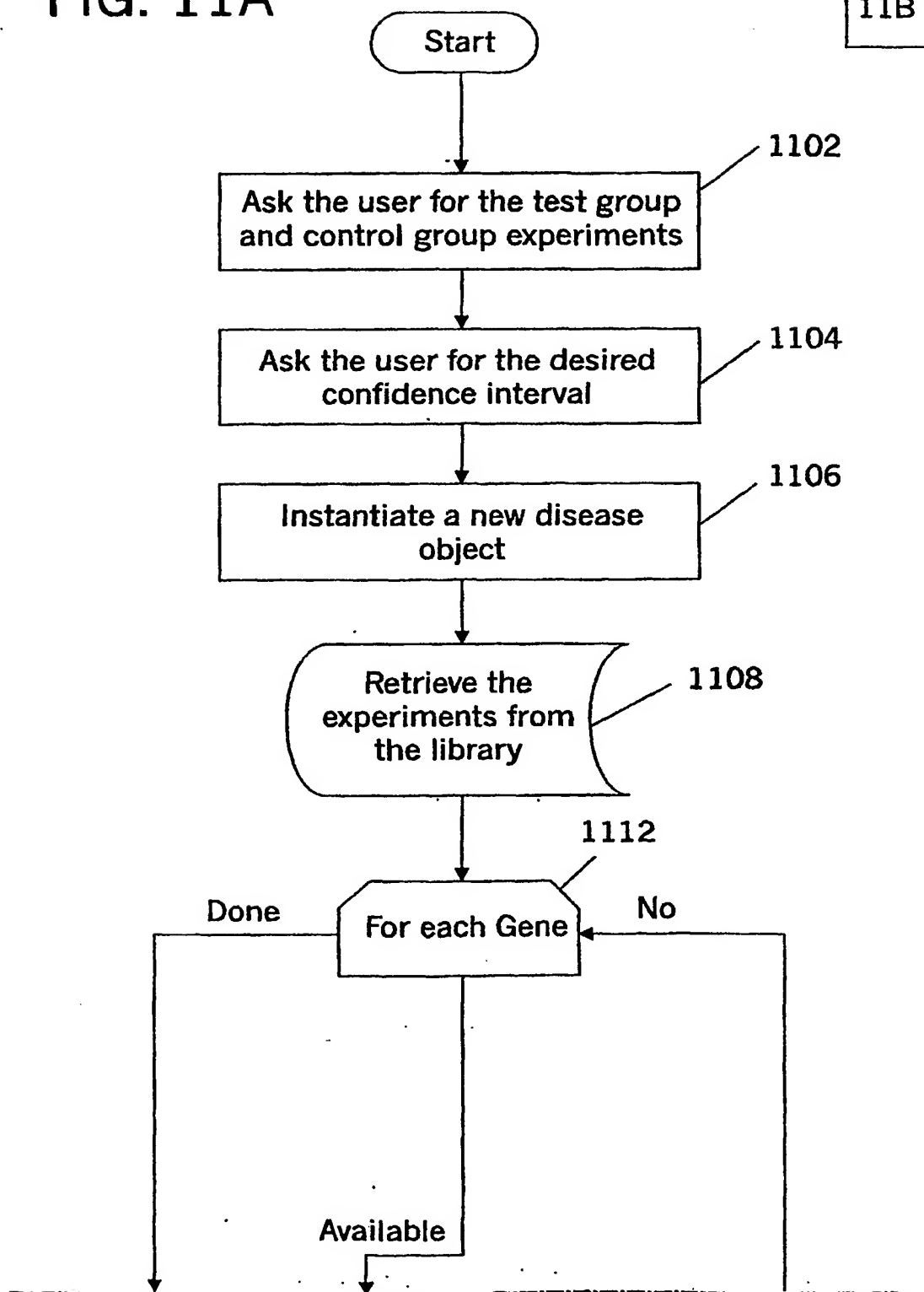


12/26

FIG. 11

FIG.
11AFIG.
11B

FIG. 11A



13/26

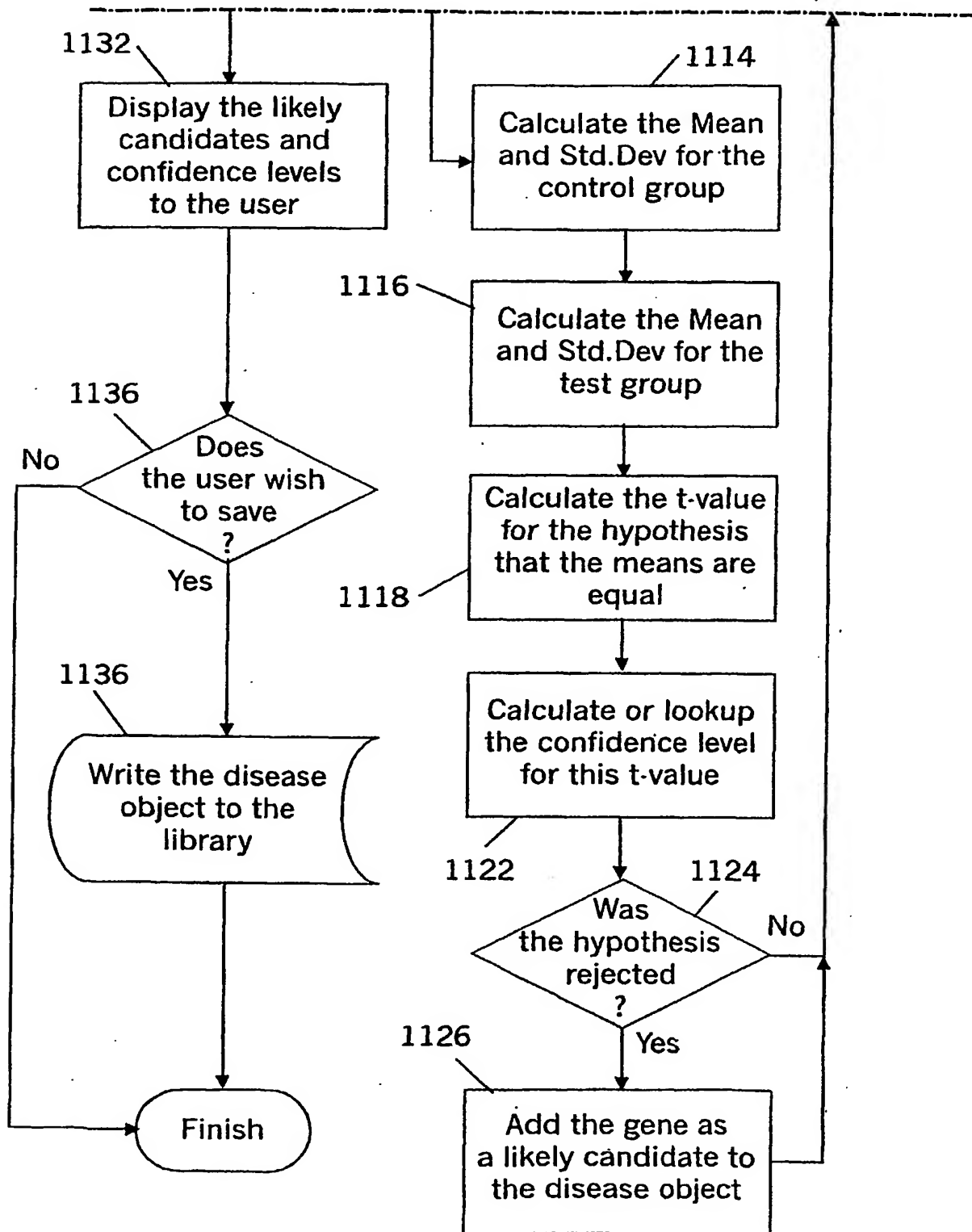


FIG. 11B

FIG. 12

14/26

Experiments	Analysis	Views
<p>Choose Experiments to Analyze</p> <div><input type="checkbox"/> Array Experiments <input type="checkbox"/> Public <input type="checkbox"/> admin/personal ▶ HL60 (14) ▶ HL60 (11) ▶ HL60 (12) ▶ HL60 (13) <input type="checkbox"/> Array/Exp Group</div> <div>Add New Experiment</div>	<p>Choose Analysis Method</p> <div><input type="radio"/> View all Genes <input type="radio"/> View marked Genes <input type="radio"/> Find Expression Patterns <input type="radio"/> Pathway Viewer <input checked="" type="radio"/> Diagnosis Engine</div> <div>Gene Marking</div>	<div>Control Group</div> <div>▶ HL60 (14) ▶ HL60 (11)</div> <div>Move to Test</div> <div>Test Group</div> <div>▶ HL60 (12) ▶ HL60 (13)</div> <div>Confidence Level 95% ▼</div> <div>Diagnose</div>

15/26

FIG. 13

FIG. 13A
FIG. 13B

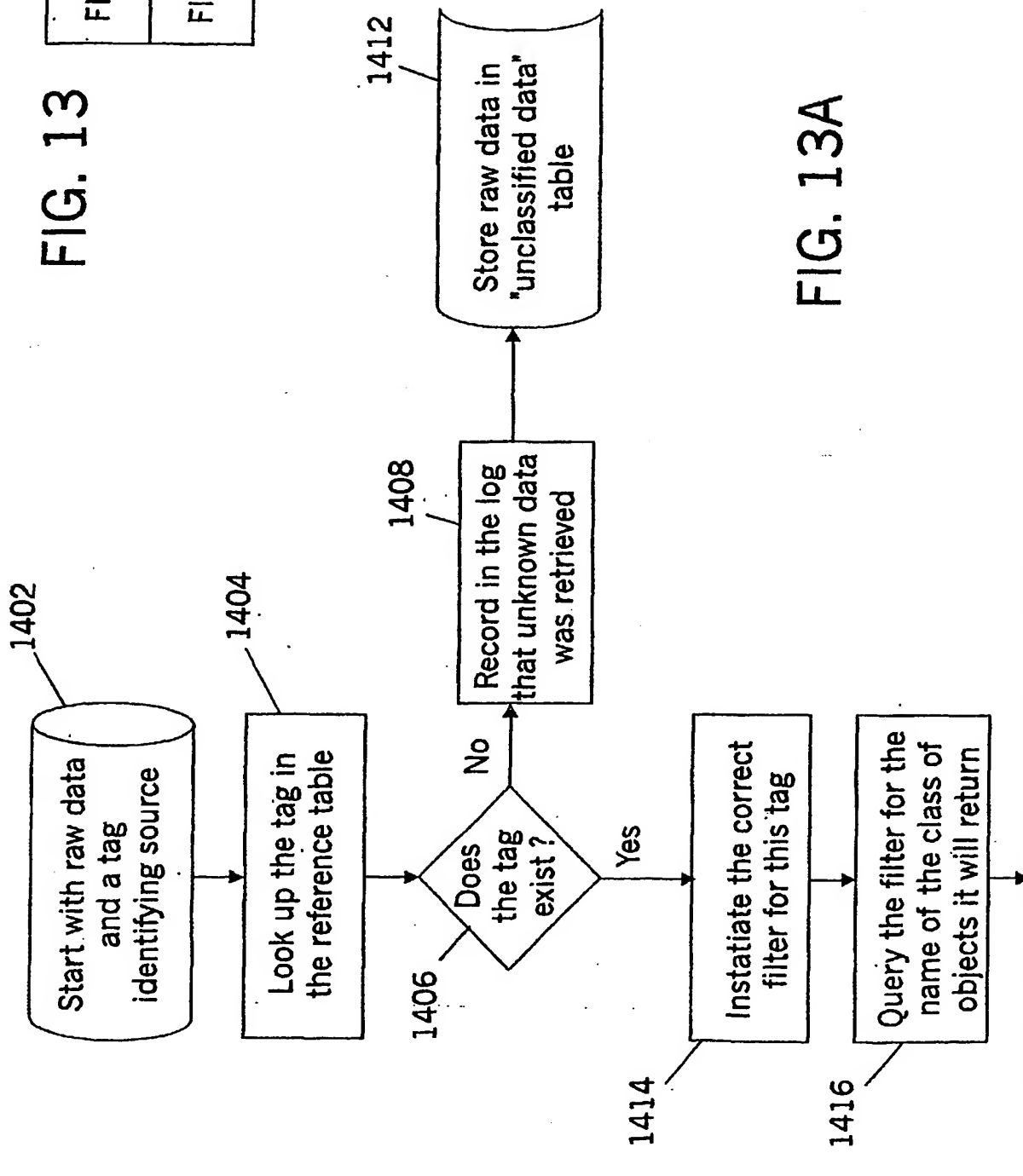


FIG. 13A

16/26

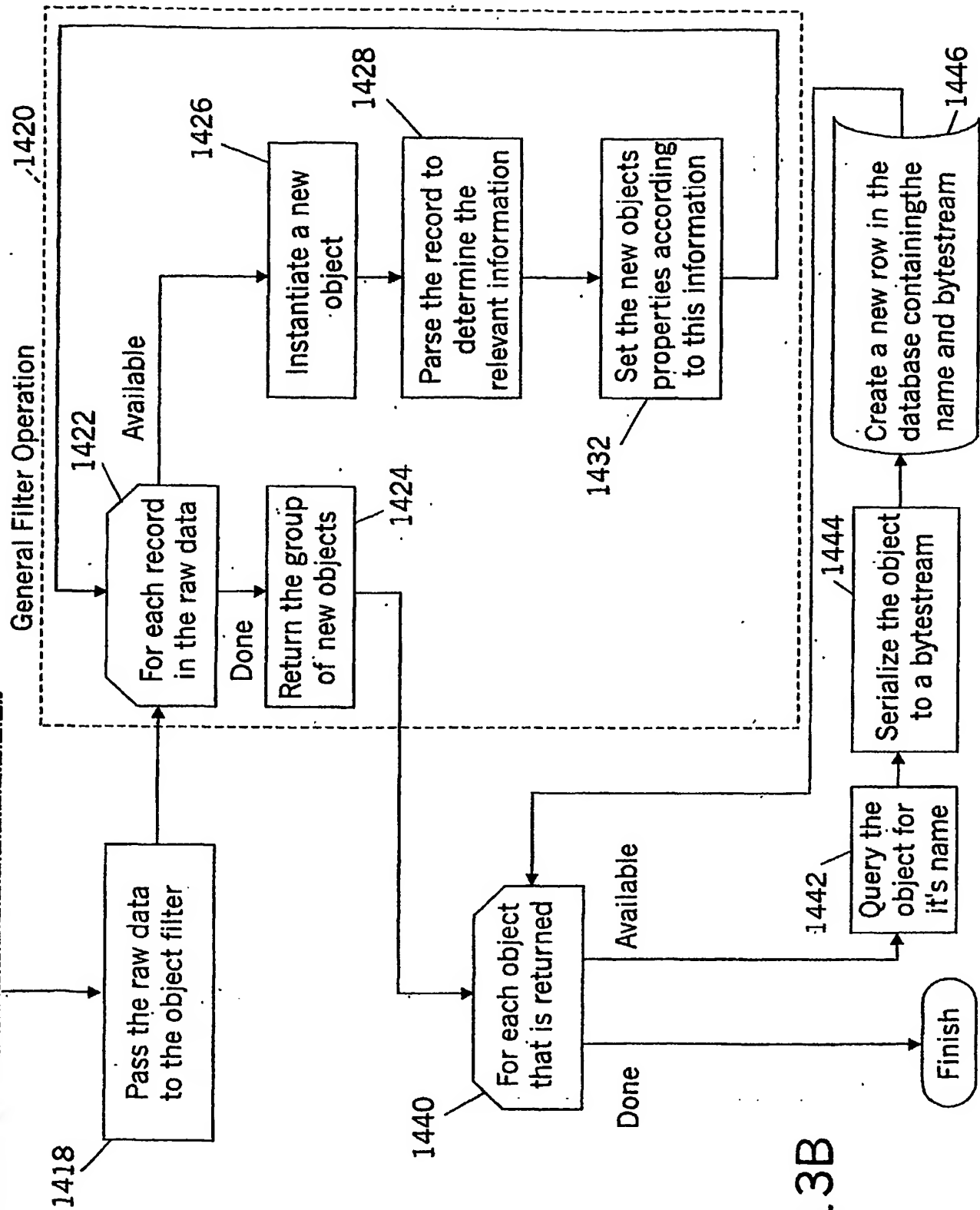
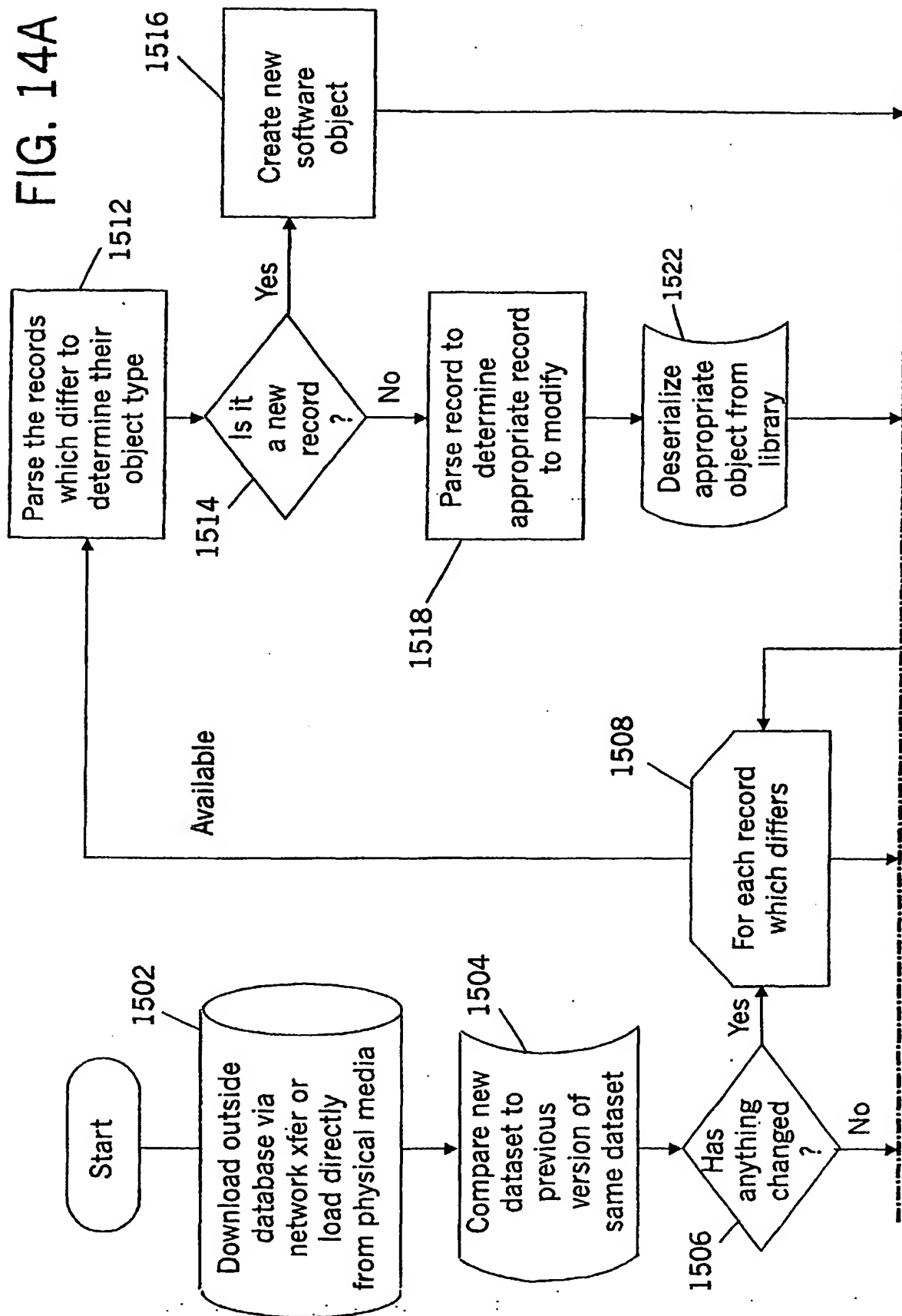


FIG. 13B

17/26

FIG. 14A



18/26

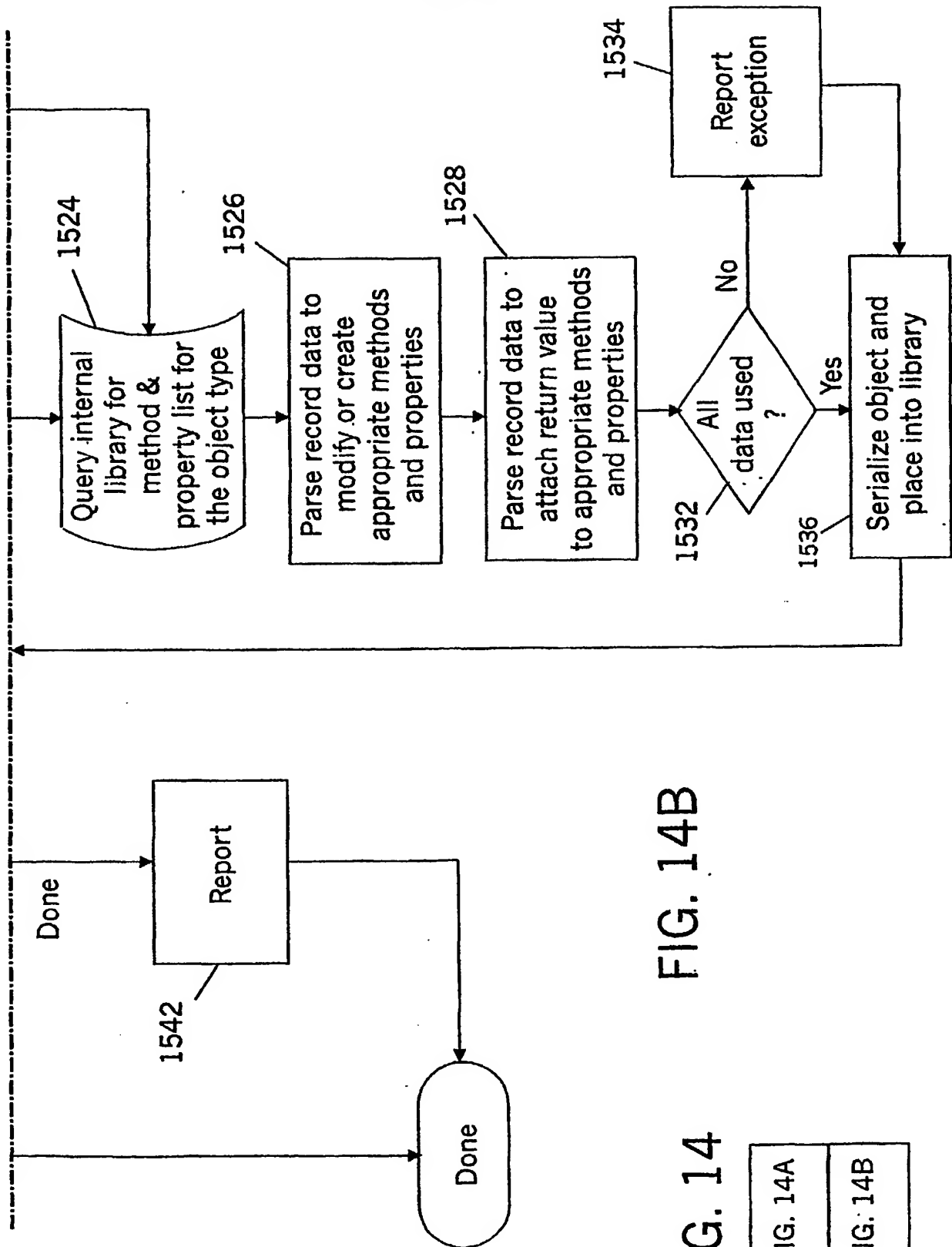
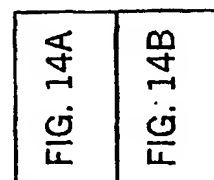


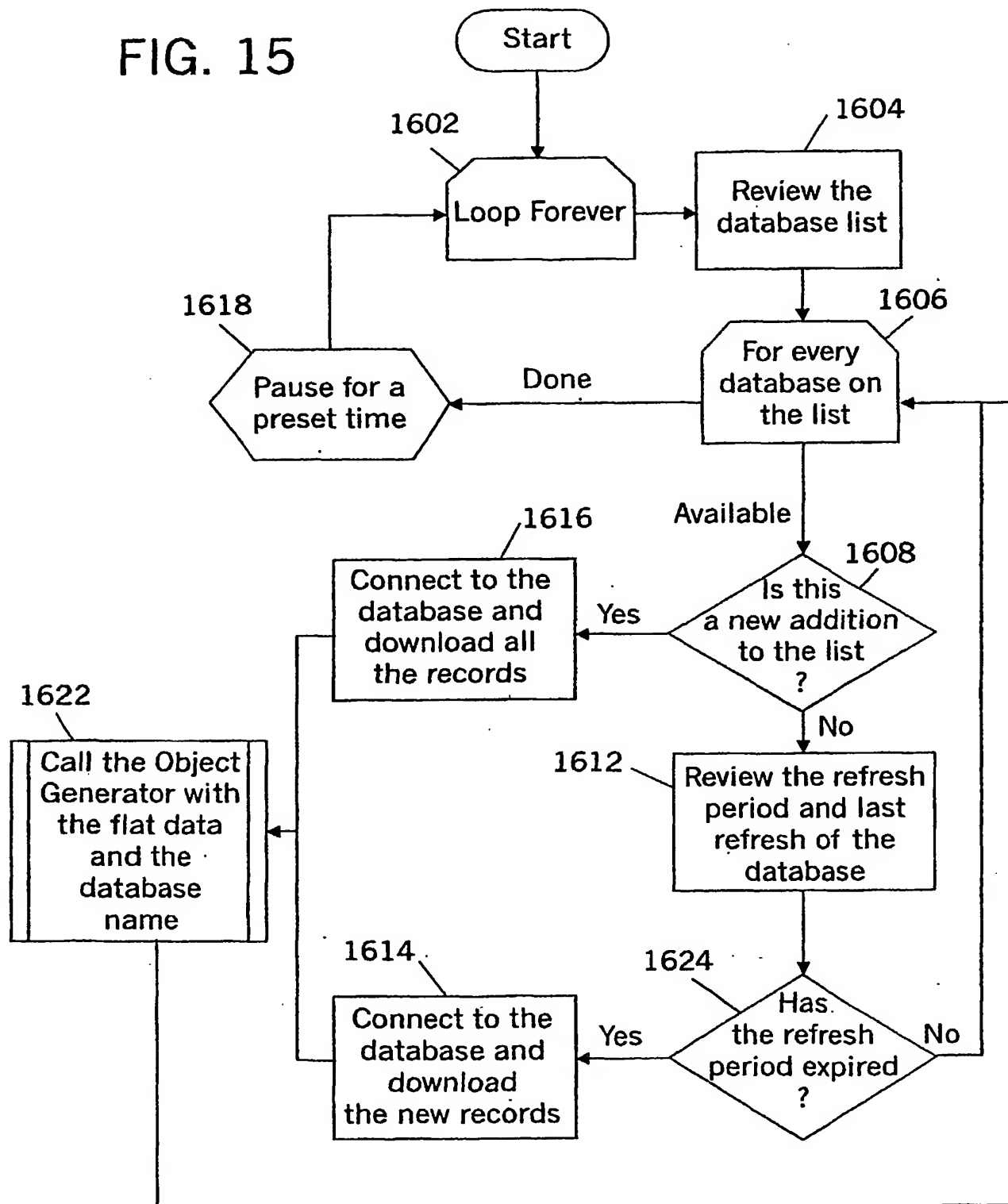
FIG. 14B

FIG. 14



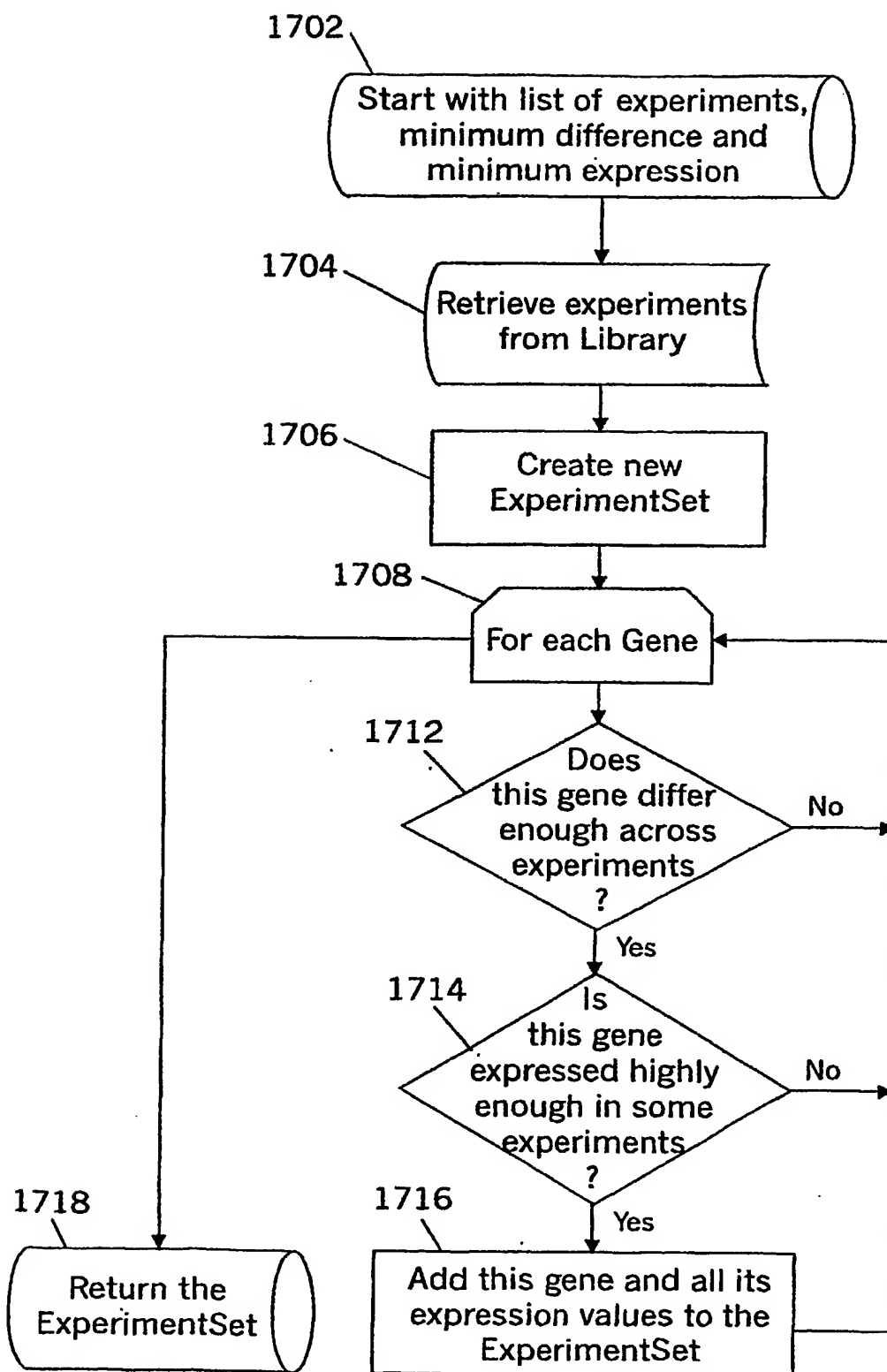
19/26

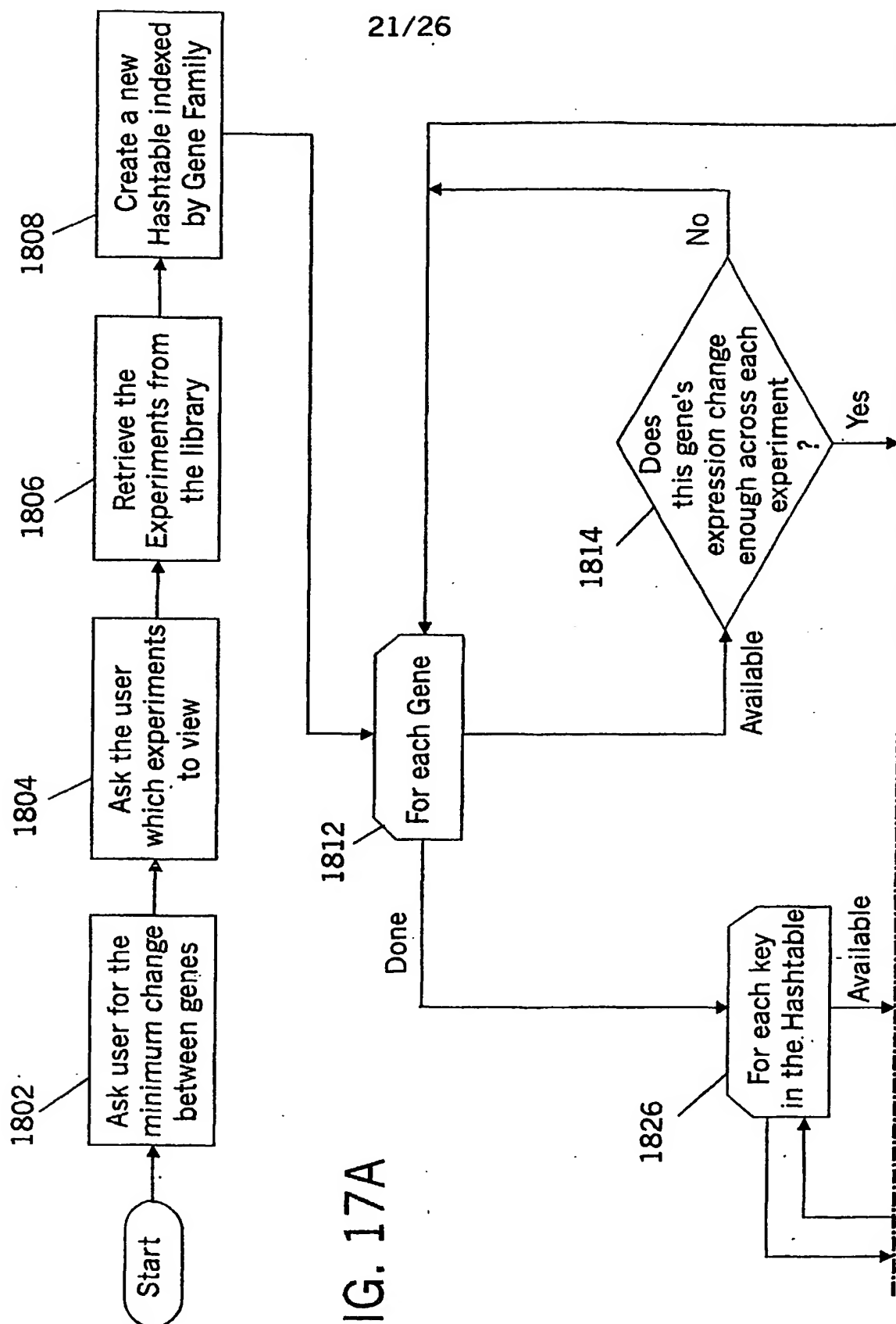
FIG. 15



20/26

FIG. 16





22/26

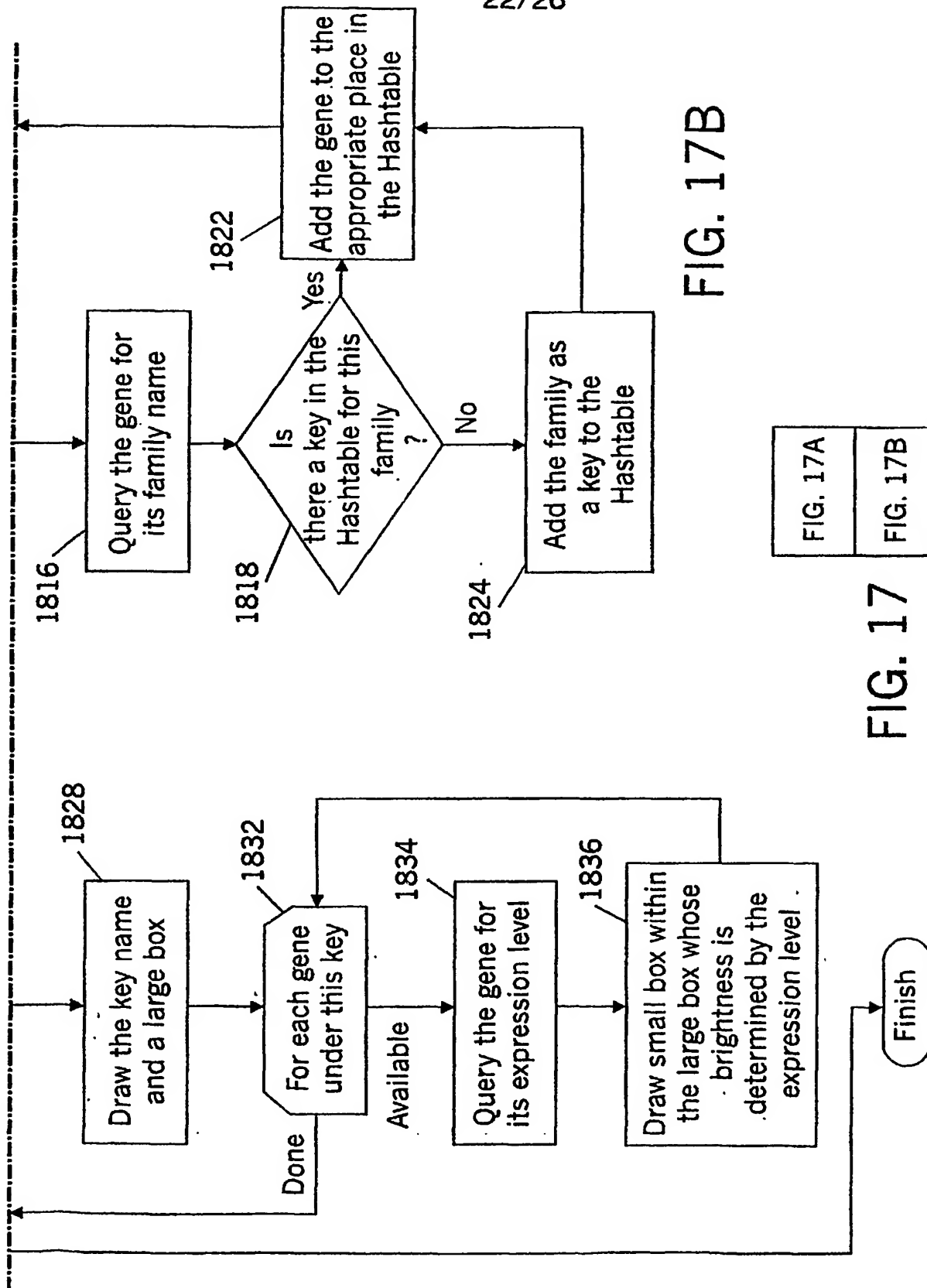


FIG. 17B

FIG. 17

FIG. 17A

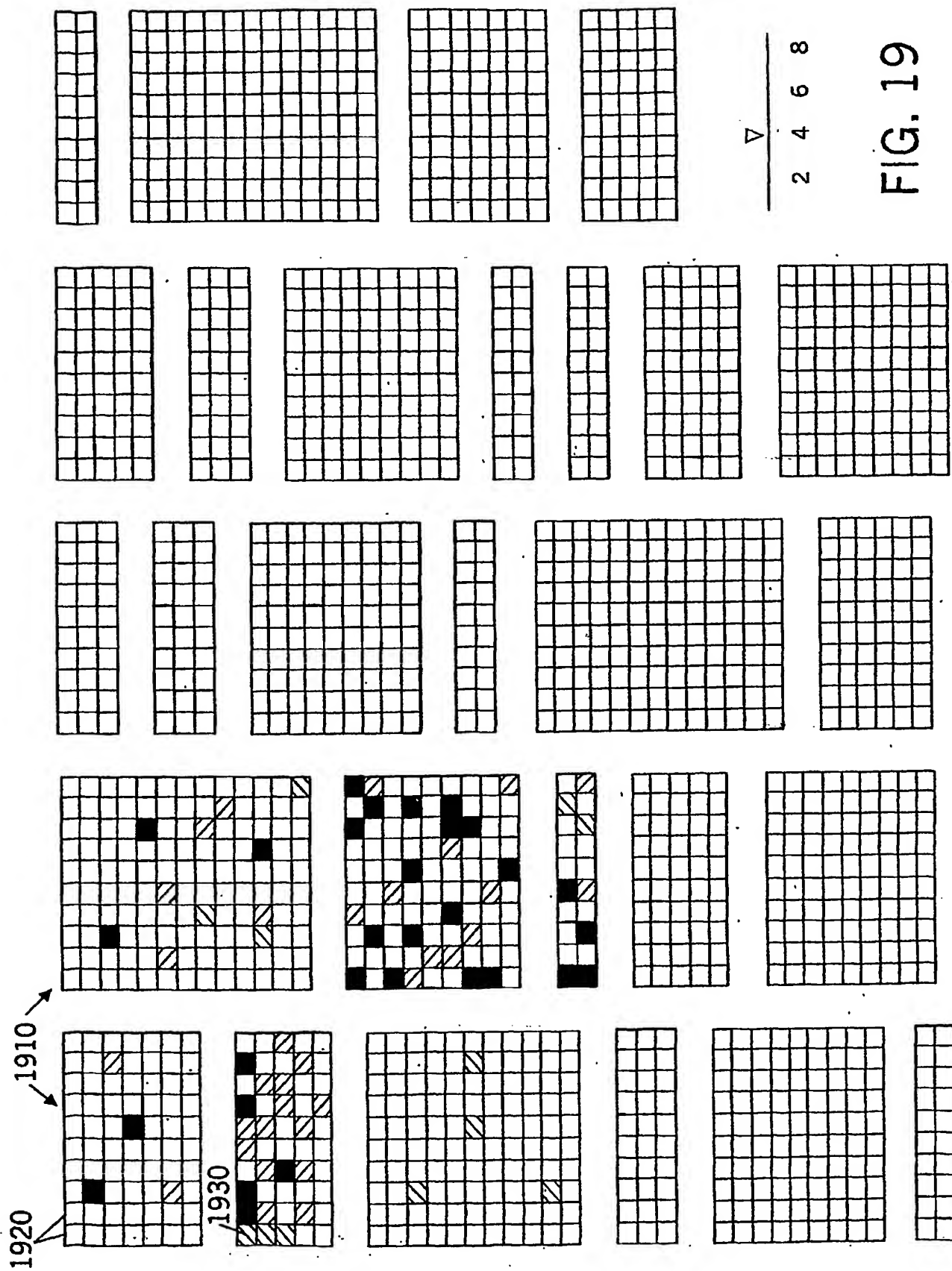
FIG. 17B

23/26

FIG. 18

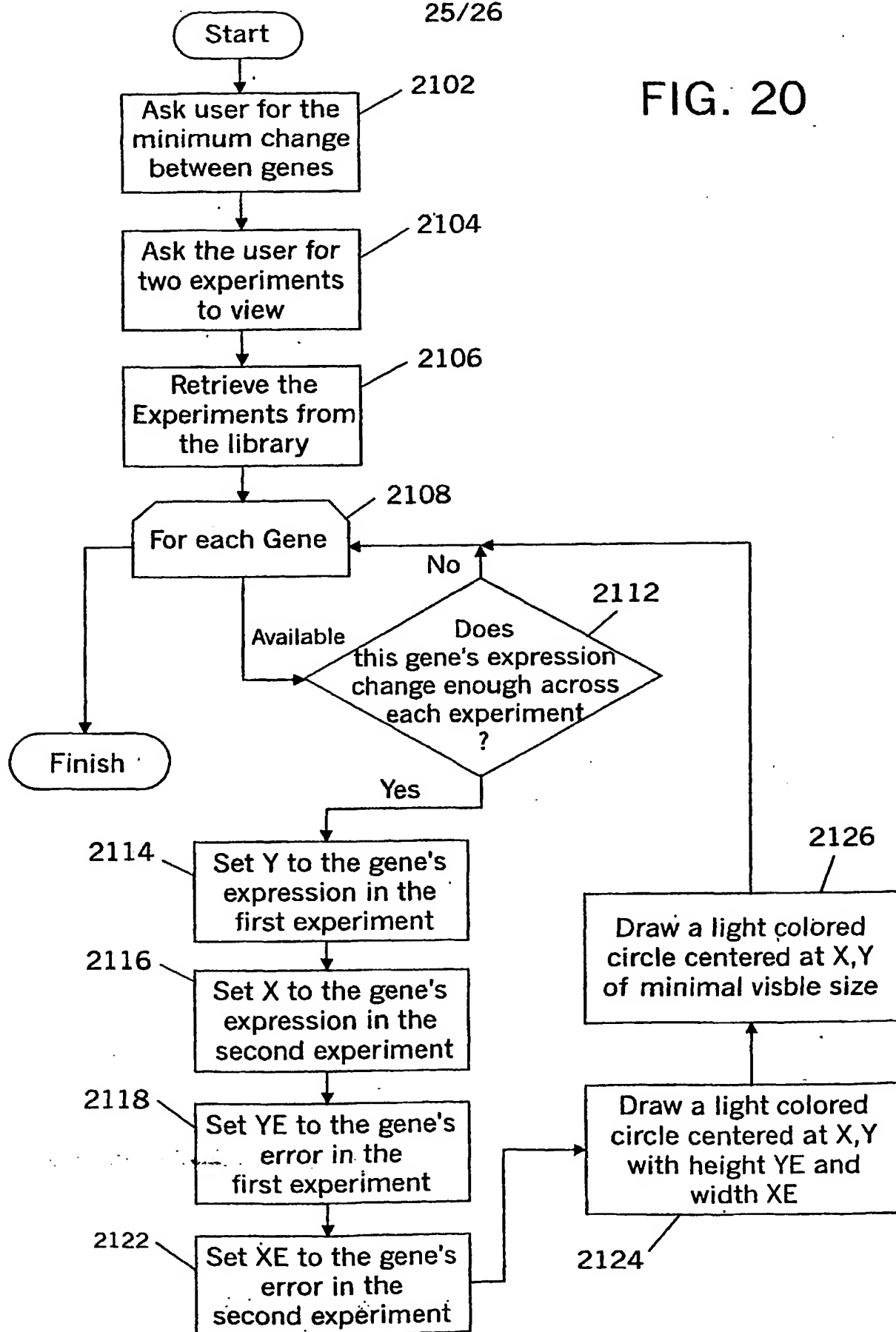
Experiments	Analysis	Views
<p>Choose Experiments to Analyze</p> <div> <input type="checkbox"/> Array Experiments <input type="checkbox"/> Public <input type="checkbox"/> admin/personal <div> <input checked="" type="checkbox"/> HL60 (14) <input checked="" type="checkbox"/> HL60 (11) <input checked="" type="checkbox"/> HL60 (12) <input checked="" type="checkbox"/> HL60 (13) </div> <input type="checkbox"/> ArrayEx Group </div> <p>Add New Experiment</p>	<p>Choose Analysis Method</p> <div> <input checked="" type="radio"/> View all Genes <input type="radio"/> View Marked Genes <input type="radio"/> Find Expression Patterns <input type="radio"/> Pathway Viewer <input type="radio"/> Diagnosis Engine </div> <p>Gene Marking</p> <p>Normalize Data</p> <div> <input type="checkbox"/> Experiment to Self <input type="checkbox"/> Genes to Background <input type="checkbox"/> Control <input type="checkbox"/> Use Log-values </div>	<p>Choose view(s)</p> <div> <input type="checkbox"/> Delta Chart <input type="checkbox"/> Series Chart <input type="checkbox"/> Scatter Plot <input type="checkbox"/> Gene Family View <input type="checkbox"/> Population Chart </div> <p>Show only genes which exceed background by: 2.0x ▼</p> <p>Show only genes which differ in expression by: 50% ▼</p> <p>Display</p>

24/26



25/26

FIG. 20



26/26

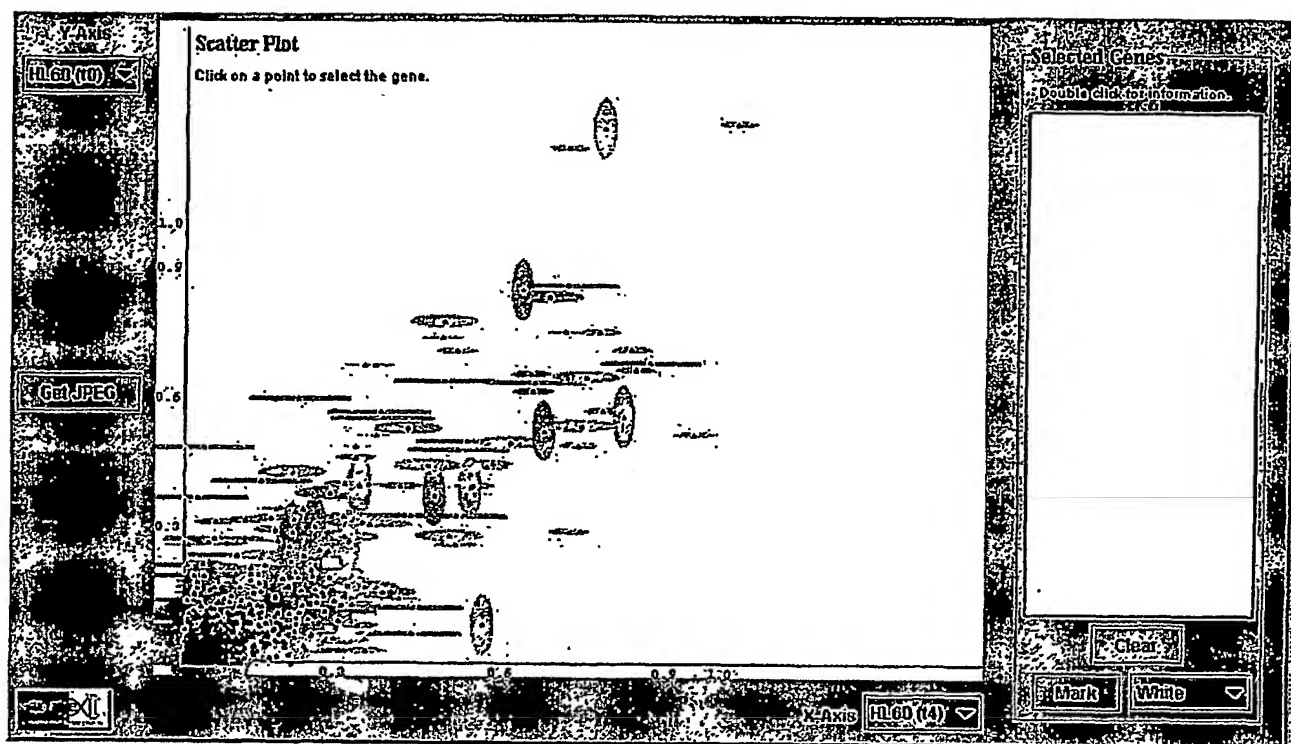


FIGURE 21